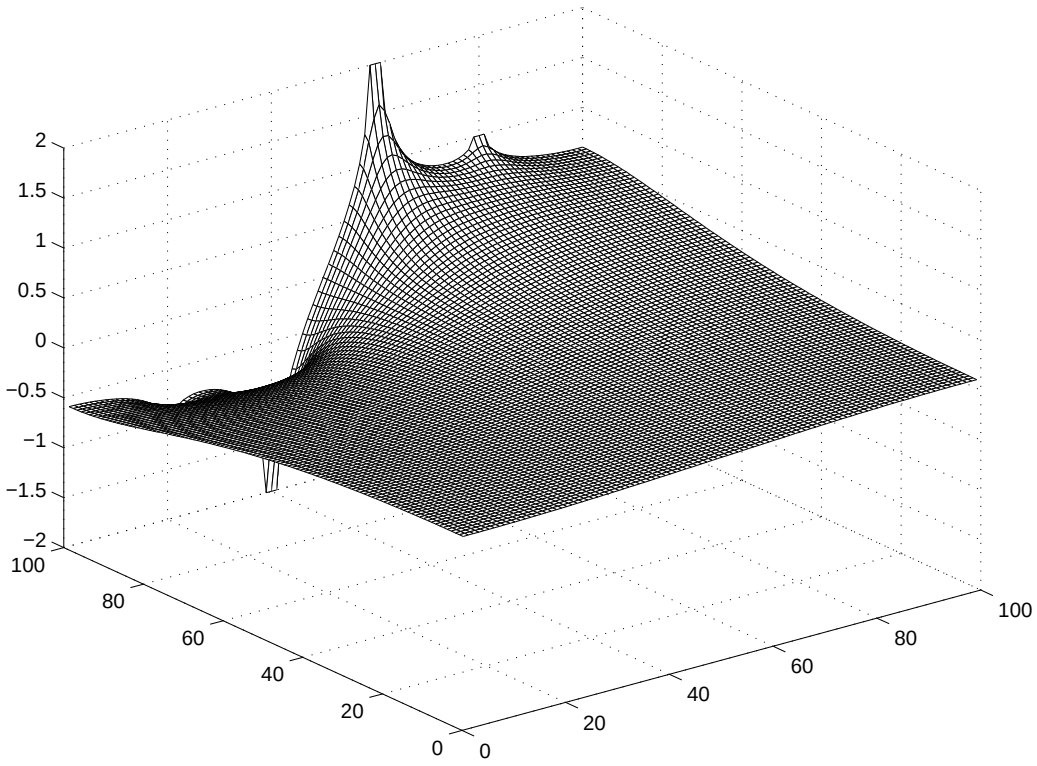
Thesis for the degree Candidatum Scientarum
Electric Impedance and Sensitivity Simulations

Filip Nicolaisen

November 8, 2004



Preface

For several 100 years now, the basic physical laws behind electro-magnetic theory have been known. It has also been known that some biological processes are of an electric nature, and one has through the years been able to apply electro-magnetic theory to biological phenomena, in and outside the human body. We have also recently begun to add the tools of computer technology to that of our field. This has led to enormous advances in medical technology, diagnostics and treatment.

In this thesis, I will address the so called *forward* problem of electromagnetics. That is, given a setup of electrodes on a conductive material, can we compute where the current will pass through? More specifically, from where will most of our signal come in a measuring experiment?

For this purpose, I have done simulations which, given suitable input parameters of the medium to be examined and the electrode configuration, will simulate the current and potential distribution of the electric signal.

The illustration on the cover is the first simulation result, the potential from a four-electrode system.

Acknowledgments

The chapter on AC-DC currents, and also much inspiration, is taken from my predecessor Vegeir Knudsen's thesis[1] I also wish to thank the people at Comsol, and especially Tore Bjørnerå, for their support in the Matlab and Femlab department.

Most physics consists of, and depends on, building an intuition for the statement of problems (as in "what is the problem?") and and an intuition for what the solution might be. The steps between these two phases is just problem-solving, which can be done in many ways, and this is what this thesis is about, trying to get to the solutions in new ways. It has still been, however dependent on the mentioned intuitions, of which I had very few when I started. Fortunately I have been able to rest on the sizeable intuitions of my tutors, Professors Sverre Grimnes and Ørjan Martinsen at the Bio-Impedance group at the university of Oslo.

There is no question that modern physics is mostly teamwork. The help and guidance from my teachers, co-students, administrative personnel and friends has been invaluable.

Finally, none of this would have been possible without family. A special thanks to my mother, who proofread most of the thesis, and to my brother, who made the last year liveable, enjoyable and affordable.

Introduction and Goals

The objectives of the thesis were the following:

1. Explore the use of the Finite-Element method for computing Electric fields, with special focus of the program Femlab

2. Investigate several problems in the impedance-measurement field

The thesis is built up by four parts. In part I, I will briefly repeat basic electro-magnetic theory, as well as stating formally the Reciprocity theorem and the Sensitivity theory. I will also derive the expressions for two analytic test-cases. In part two I will do a comparative presentation of some different programming environments. In part three I will investigate some problems through simulations. Part four is the conclusion.

Contents

Part I
Theory

Much of the known theory is taken from selected textbooks, especially [5] and [2]. Where other material is used, this is cited appropriately.

1 Syntax

In this paper, vectors will be written with bold font, as in $\mathbf{v}$. Vector fields will be written in the same, only in capitol letters, as in $\mathbf{E}$. A scalar field will be symbolized by a capital letter. A vector may indicate that we are dealing with a complex number, but this will be clear from the context. If a complex number is used, a symbol with a single prime is a real component, double primed is the imaginary part. A list of symbols can be found in the appendix.

2 Electromagnetic Foundation

A brief introduction to the basic electromagnetic theory is given. This will serve two purposes, firstly, it will be extended with the theory of electric sensitivity and reciprocity. Second, in the discussion of results, many of the relations between the electric field, electric flux, potential, current, current density etc will be used.

2.1 Electric Charge, Insulators, Capacitors and Coulomb's Law

Electric Charge is a fundamental quantity, like mass and length. Charge can be positive or negative, where similar charges repulse each other and opposite charges attract. The SI base unit for electric charge is the Coulomb (C), which is an old unit whose size does not correspond directly to any quantity in nature, and is in truth a very large charge. Two items separated by one meter, each with 1 C positive charge, will repel each other with a force of approximately 9 000 000 000 Newton, enough to lift more than a thousand Eiffel Towers.
A more appropriate unit charge is the charge of one electron (e), which is approximately $1.6 \times 10^{-19} C$. The electron is the charge carrier for most electric applications, although ions are sometimes used, and are typical for

bioelectricity, electricity in biological systems. An ion is an atom which has excess electrons (negatively charged) or too few (positively charged).
An *electric conductor* is a material with very high electric conductivity, usually a metal like copper or silver. Having a large conductivity will mean that the conductor has a large number of free electrons that can "slide along" energy bands in the molecule grid. An *electric insulator* is the opposite, where there are few or no free electrons (or ions).
Coulomb's Law states that the force attracting or repelling two electrically charged points falls as the square of the distance between them. It has been experimentally measured to be

$$F = \frac{1}{4\pi\epsilon_0}\frac{|q_1 q_2|}{r^2}$$

where q are the charges, ϵ_0 is the permittivity of free space, a fundamental constant in nature. $\frac{1}{4\pi\epsilon_0}$ is approximately $9 \times 10^9 N \cdot m^2/C^2$. The force is on q_2 from q_1, and is directed away from q_1. This is the same as the force on 1 from 2, in the opposite direction, because of Newton's third law. An illustration is included in figure 2.1.

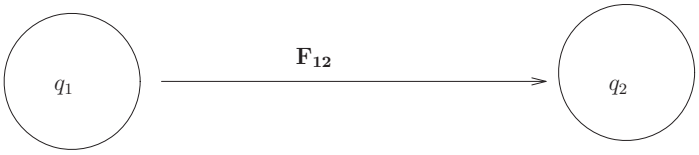


Figure 2.1: The electric force between two charges

Note that this force is correct only in vacuum (or approximately in air). If we are dealing with a solid (insulator), we need to modify the permittivity with a *relative* permittivity, ϵ_r. The force then becomes

$$F = \frac{1}{4\pi\epsilon}\frac{|q_1 q_2|}{r^2}$$

where

$$\epsilon = \epsilon_0\epsilon_r$$

2.2 Electric Field, Gauss's Law and Electric Potential

Like the gravitational field is the acceleration the gravitational force causes per unit mass, the *electric field* is the electric force on a particle per unit

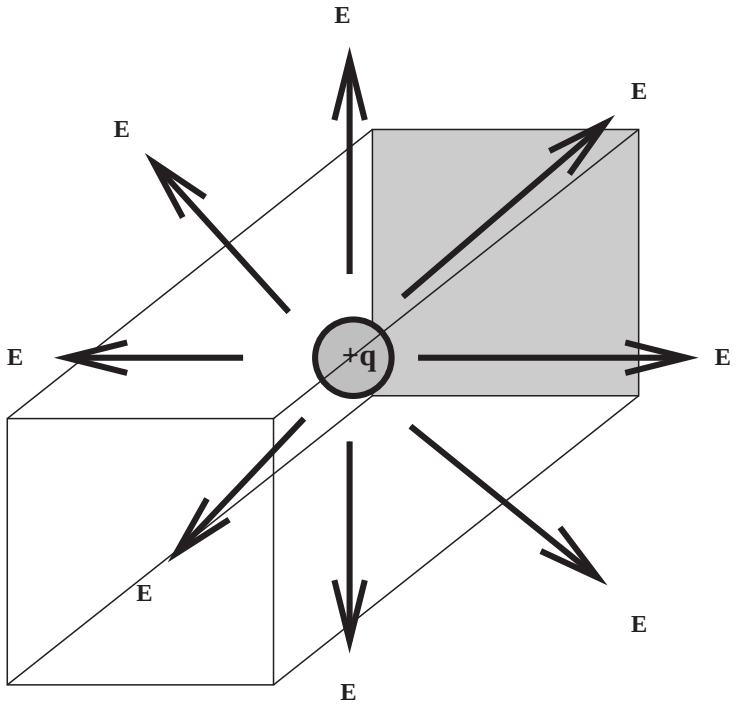


Figure 2.2: A figure demonstrating Gauss's law, the flux through a closed surface is proportional to the charge enclosed by the surface

positive charge, so that we have

$$\mathbf{E} = \frac{\mathbf{F}}{q_0}$$

directed away from a positive source.
Gauss's law states that the total electric flux (electric field times the area) through a closed surface is proportional to the charge enclosed, or

$$\Phi_E = \oint \mathbf{E} \cdot d\mathbf{A} = \frac{q}{\epsilon}$$

as depicted on figure 2.2. If a charge is placed outside the surface, the flux in on out through one side will be equal and opposite to the flux on the opposite side.

This will later be mentioned as one of Maxwell's equations.
The *electric potential energy* is the work needed to move an electric particle through an electric field. The electric force is a conservative force, so that the work done in moving from a point a to point b is independent of the path

taken. We can then also express the work done as a potential energy U, so that

$$W_{a\to b} = \int_a^b \mathbf{F} \cdot d\mathbf{l} = U_a - U_b = -(U_b - U_a) = -\nabla U$$

Thus, the work required to move a particle q_2 in a field from a particle q_1 between two points a and b will be

$$W_{a\to b} = \frac{q_1 q_2}{4\pi\epsilon_0}\left(\frac{1}{a} - \frac{1}{b}\right)$$

If we place the point charge q_1 at point a, the potential energy U when a test charge q_0 is at a distance r from q_1 is

$$U = \frac{1}{4\pi\epsilon_0}\frac{q_0 q_1}{r}$$

We can then define the *electric potential* as the energy per unit charge, so that the electric potential from a point charge as

$$\Phi = \frac{U}{q_0} = \frac{1}{4\pi\epsilon_0}\frac{Q}{r}$$

The SI unit of electric potential is the volt, and is defined as 1 V = 1 volt = 1 J/C = 1 joule/coulomb. This gives rise to the expression *voltage*, which is defined as the electric potential difference between some point in an electrical circuit system and another (reference point).
The electric potential and the electric field are related by the electric potential gradient. Restating the equation for potential energy above,

$$V_a - V_b = \int_a^b \mathbf{E} \cdot d\mathbf{l}$$

We can also write this equation on differential form

$$\mathbf{E} = -\nabla V$$

2.3 Capacitance, Current and Resistance

If we place an insulator between two conductors we can make a *capacitor*, on which we can build up an electric field in the insulator. The capacitance is defines as the ratio of charge built up onthe conductors to the voltage difference,

$$C = \frac{Q}{V}$$

and has the SI unit farad,

$$1\ F = 1 \text{ farad} = 1\ C/V = 1 \text{ coulomb/volt}$$

The current density is the current per area, and is denoted J. The velocity of electrons in a standard copper wire is around 0.1 mm/s. Since the random movement velocity of electrons at room temperature in copper is around 10^6 m/s, we see that the current must be a result of the millions of electrons moved per second by the electric field.

In an ideal, or *ohmic* material, there is a linear relationship between the current density and the electric field. This is a material property, and is related to how easy it is for a charge to travel through the medium. This is the previously mentioned conductivity, σ, or the inverse resistivity, ρ , defined as

$$\sigma = \frac{1}{\rho} = \frac{\mathbf{E}}{\mathbf{J}}$$

This is called Ohm's law.

If we have a conductor with two different potentials (at two different places), the equations state that there will be an electric field between the two points. This will cause free charges in the conductor to pass along the conductor, thus giving rise to an *electric current*. The number of charges that flow through a cross-sectional area A per unit time is the current through that area.

$$I = \frac{dQ}{dt}$$

The electric *resistance* is an object property relating the potential drop over the object to the current passing through it. If we have a conductor with two endpoints at different potentials, there will be a current passing through it. For the simple setup in figure 2.3, we have the two potentials Φ_1 and Φ_2 at the left and right end, respectively.

The potential difference between Φ_1 and Φ_2 is V. If the area of the end surfaces are A, we have a uniform electric field, current and current density, and the three are related through

$$I = \mathbf{J} \cdot \mathbf{A} = \frac{\mathbf{E} \cdot \mathbf{A}}{\rho}$$

We know from before that

$$V = \int \mathbf{E} \cdot dl = EL$$

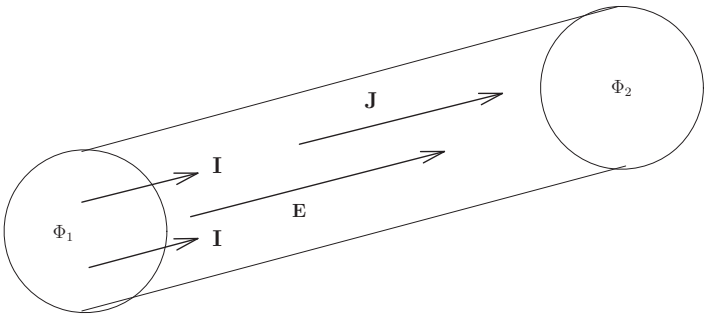


Figure 2.3: The resistance in the conductor is calculated by dividing the voltage by the potential

where L is the length of the conductor. Inserting the previous expression for E into the expression for the voltage, we have

$$V = \frac{\rho L}{A} \, I$$

The expression before I is a material property that relates the current passing through the conductor to the voltage. For a more complicated current-path, the term will be a bit more ominous. We will later return to the expression of transfer resistance (impedance), when we measure current and voltage at different ports.

2.4 Displacement Currents and Ampere's Law

We return to the capacitor. Picture the basic capacitance setup, illustrated in figure 2.4 (The example is taken from Bioimpdance Basics [2])

If we apply a sinusoidal voltage to the circuit, the two conductor plates will be charged along with the voltage to $Q = VC$. The capacitance of a plate capacitor can be shown to be

$$C = (A/d)\epsilon$$

where ϵ is the permittivity. It plays somewhat of the same role as conductivity does for conductors. As the plates are charged, an electric field will be set up by the oppositely charged plates. If the external field over the dielectric is E, then there will flow a *displacement current* in the dielectric. We can find an expression for the charge on the conductor

$$Q = CV = \frac{\epsilon A}{d}(Ed) = \epsilon EA$$

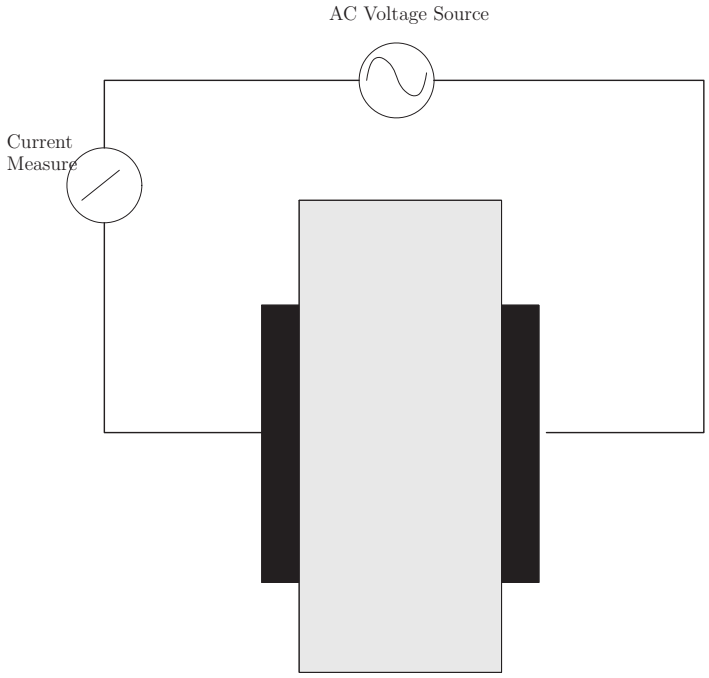


Figure 2.4: The basic capacitance experiment, two conductive plates with a dielectric between

The current through a circuit-element is $\frac{dQ}{dt}$, so that we have the displacement

$$I_d = \epsilon_s \frac{dE}{dt} A$$

In a circuit like the one in figure 2.4, the current through the conductors is 90 degrees out of phase with the displacement current through the dielectric, which means that the maximum of the currents come at a time which is 1/4 of the AC period later. If it has a resistive property as well, it will have a resistance which is in phase with the voltage and in-phase current.
Ampere's law says that a current through a closed circuit causes a magnetic field. The line-integral of this field is

$$\oint \mathbf{B} \cdot d\mathbf{l} = \mu_0 I_{encl}$$

where μ_0 is a material constant called the permeability of free space, and I_{encl} is the current enclosed in the integration path.

But we also have to include the displacement current in this expression, so that the generalized Ampere's law is

$$\oint \mathbf{B} \cdot d\mathbf{l} = \mu_0 (Iin + Idisp)$$

We now have the equations we need for our treatise.

2.5 Maxwell's Equations

Maxwell's equations sum up some of the equations we have arrived at, and a few more that we will not be needing. The first one is Gauss's law, the second is the corresponding law for magnetic fields, which states that a gauss-surface around a magnetic source will always have zero net flux, since there are no magnetic mono-poles like there are electric charges.

The third law is Faraday's law, that states that a time-changing magnetic field causes a rotational electric field. We will not be using this law. The fourth law is Ampere's law just mentioned above.

$$\begin{aligned}
\oint_S \mathbf{E} \cdot d\mathbf{A} &= \frac{Q_F}{\epsilon_0} \\
\oint_S \mathbf{B} \cdot d\mathbf{A} &= 0 \\
\oint \mathbf{E} \cdot d\mathbf{s} &= -\frac{dB}{dt} A \\
\oint \mathbf{B} \cdot d\mathbf{s} &= \mu_0 I + \mu_0 \epsilon_0 \frac{dE}{dt} A
\end{aligned}$$

These can also be written in differential form

$$\begin{aligned}
\nabla \cdot \mathbf{D} &= \rho_F \\
\nabla \cdot \mathbf{B} &= 0 \\
\nabla \times \mathbf{E} &= -\frac{\partial \mathbf{B}}{\partial t} \\
\nabla \times \mathbf{H} &= \mathbf{J_F} + \frac{\partial \mathbf{D}}{\partial t} max4
\end{aligned}$$

Here the subscript F indicates free charges, as those found in conductors. D is related to E by $D = \epsilon E$, and H relates to B by $B = \mu H$.

3 DC- and AC-currents and the Laplace-Equation

If we take the divergence of Maxwell's fourth equation (**??**) we end up with a homogenous equation, since the divergence of a curl is zero. In the DC-case

there can't be any time dependency in the electric field, so the time derivative of $\mathbf{D}$ must also be nil. This leaves us with the simple equation

$$\nabla \cdot \mathbf{J_F} = 0.$$

Combining this equation with Ohm's law and the definition of the potential gives us the differential equation

$$\begin{aligned} \mathbf{J} &= \rho\, \mathbf{E} \\ \mathbf{E} &= -\ \nabla\ \Phi \\ &\Downarrow \\ -\nabla \cdot (\rho \nabla \Phi) &= 0 \end{aligned}$$

the Laplace equation. This is the equation we are generally concerned with for solving stationary systems. Solving for electric current density will amount to finding the divergence of the field. For AC-currents the picture is slightly more complicated. In addition to the conductive current, we also have a capacitive term. Now the time-derivative will not be zero.

$$\nabla \cdot (\mathbf{J}_F + \frac{\partial \mathbf{D}}{\partial t}) = 0$$

These are the constitutive equations that we will be solving for in the simulations to come.

3.1 Permittivity, Conductivity, Resistivity and Complex values

If a material has both conductive and dielectric properties, as is common in biological tissue, it can be useful to relate them to each other. Since displacement current will generally be leading the in-phase current by a phase determined by the angular frequency, we can define complex conductivities and permittivities so that this property is conserved. Their relationships are given in table 1, and are taken from [2]. The complex conductivity can be defined as

$$\sigma \equiv \sigma' + i\sigma''$$

$$\epsilon \equiv \epsilon' - i\epsilon''$$

Using these relations, we can find an alternative expression for the complex conductivity,

$$\sigma = \sigma' + \mathrm{j}\ \omega\epsilon'$$

Table 1: Table of the relationships between real and complex values of conductivity, permittivity and resistivity

$$\begin{aligned}
\sigma &= i\omega\epsilon \\
\sigma'' &= \omega\epsilon' \\
\sigma' &= \omega\epsilon'' \\
\epsilon'' &= \sigma'/\epsilon \\
\epsilon' &= \sigma''/\omega \\
\rho'' &= \sigma''/|\sigma|^2 \\
\rho &= \sigma'/|\sigma|^2
\end{aligned}$$

Transfer Impedance

We will define the expression *Transfer Impedance* as where one measures voltage and current at different ports. In the figure 3.1 a model for three- and four electrode transfer impedance is demonstrated.

In both cases, the transfer impedance, Z, is

$$Z = \frac{\Delta V}{I}$$

where ΔV is the potential-difference as can be measured by the Voltmeter in the setup, and I is the current as can be measured by the Amperemeter.

4 Sensitivity

Sensitivity is a concept that tries to describe what effects local changes in resistivity will have on the total impedance. It can be defined in several ways. One is by *volume sensitivity* described in Basics[2]. This is the ratio of the conductance contribution of a small, defined volume at two different points, $\mathbf{r}_1$ and $\mathbf{r}_2$.

We also have a more formal definition, where local sensitivity is a dot product of a measuring current-density, $\mathbf{J}_{CC}(\mathbf{r})$ and a so-called reciprocal current-density, $\mathbf{J}_{rec}(\mathbf{r})$.

$$S(\mathbf{r}) = \frac{\mathbf{J}_{cc}(\mathbf{r}) \cdot \mathbf{J}_{rec}(\mathbf{r})}{I^2}$$

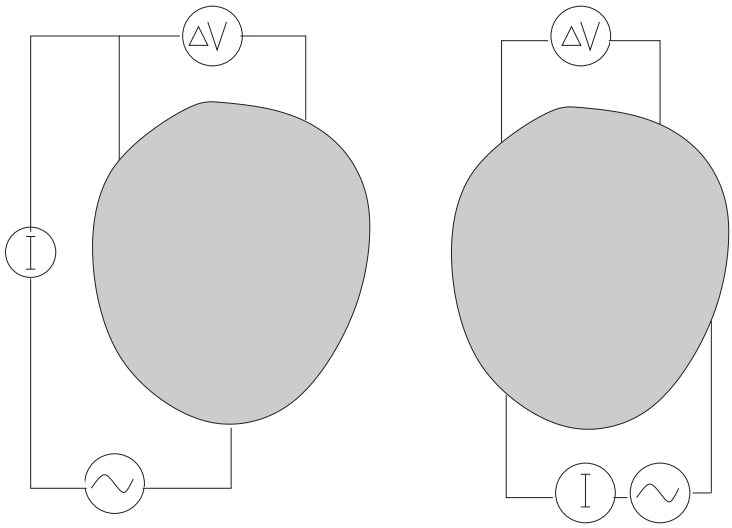


Figure 3.1: Transfer impedance in a three- and four-electrode setup

The reciprocal current is an imagined unit current through the pickup electrodes. An illustration of a typical setup is given in figure 4.1.

Geselowitz [4] gives the change in impedance in an isotropic volume conductor as

$$\Delta Z = \int_a \Delta\rho \; S dv$$

where $\Delta\rho$ is the change in resistivity and a is the volume of the piece where the resistivity changes. The baseline-impedance, which is the measured impedance before an increase or decrease in local conductivity, is equal to

$$Z_0 = \rho_0 \int_V S \; dV$$

where V is the total volume in the conductor.

So, looking at figure 4.1, if the resistivity in area a increases 5 %, and the sensitivity in that area is some value S_1, the change in the impedance will be

$$\Delta Z = S_1 \Delta\rho \; = S_1 \cdot 0.05\%$$

The relative change in the total impedance in the system is then

$$\frac{\Delta Z}{Z_0} = \frac{S_1 \; 0.05\%}{\rho_0 \int_V \mathbf{J}_l oc \cdot \mathbf{J}_m \; dV}$$

If we now look at the shaded area in the middle of figure 4.1, and imagine this area to have a base resistivity $\rho' >> \rho_0$. From the equations above,

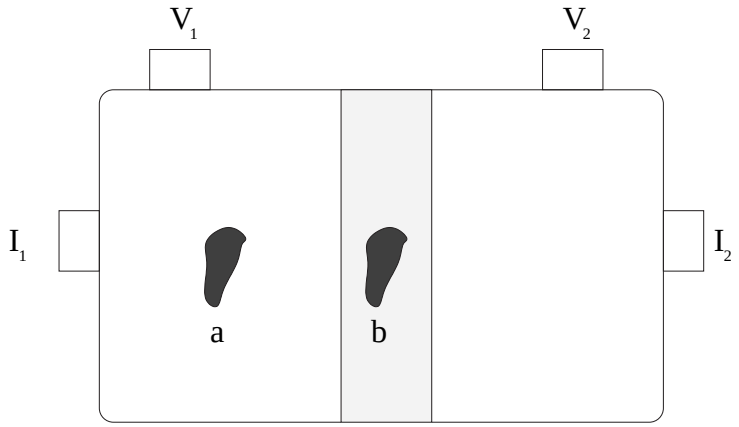


Figure 4.1: A typical experiment for impedance computation. The current carrying electrodes induce a current which result in a potential that can be measured by the pickup-electrodes. The regions a and b will have a local change in resistivity, $\Delta\rho$

we see that that a local change in resistivity of 5 % in region b has a much greater effect on the total impedance of the conductor. We can therefore introduce a term called *Weighted Sensitivity*, which is the sensitivity field multiplied by the resistivity, and will be noted as $\mathfrak{S}$.

$$\mathfrak{S} = \rho S$$

This thesis does not concern itself remarkably with different materials, except from the difference in electrodes and conductors, and we are not particularly interested in what is going inside the electrode. It is, however, an interesting field for further study.

5 The Reciprocity Theorem

We can use figure 4.1 to illustrate the Reciprocity Theorem for electric fields. Briefly, this theorem states that it does not make any difference if you use the pickup electrodes as current-carrying electrodes or vice versa.

More formally, we say that the transfer impedance using the current-carrying electrodes as source and the pickup-electrodes for listening, is the same as the transfer impedance we would have got if we sent a current through the pickup electrodes, and listened at the current-carrying electrodes.

Say that we call the potential set up by the current-carrying electrodes Φ, and the hypothetical potential set up by a current in the pickup-electrodes

Ψ_{PP}. We then call the potential difference setup up by Φ between the *pickup-electrodes* $\Phi(CC)$. Similarly, we call the potential difference set up by Ψ between the *current-carrying* electrodes $\Psi(PP)$.

In other words, the reciprocity statement says that

$$\frac{\Psi_{CC}(PP)}{I_{CC}} = \frac{\Phi_{PP}(CC)}{I_{PP}} reciprot heorem \tag{1}$$

We will quickly show how we arrive at this result (from Geselowitz [4]) As stated, Φ and Ψ are the potential fields set up by the current-carrying and pickup electrodes, respectively. We can the apply Green's Theorem from vector calculus

$$\int\int_S \mathbf{F} \cdot \mathbf{n}\ dS = \int\int\int_V \nabla \mathbf{F}\ dV green \tag{2}$$

where $\mathbf{n}$ is the unit vector normal to the surface, S is over the entire surface and V is over the entire volume of the conductor (or simplicity we will assume constant conductivity). We will use shorthand notation for the surface-integrals, so that $\int\int_S \ldots dS = \int_S$ and similarly for the volume-integrals.

If we now set the vector field $F = \Psi\nabla\Phi$, we get

$$\int_S \Psi\nabla\Phi\ dS = \int_V \nabla\Psi\nabla^2\Phi\ dV + \int_V \nabla\Psi\nabla\Phi\ dV green_1 \tag{3}$$

known as Green's first identity. We can do the same for $\mathbf{F} = \Phi\nabla\Psi$.

$$\int_S \Phi\nabla\Psi\ dS = \int_V \nabla\Phi\nabla^2\Psi\ dV\ + \int_V \nabla\Phi\nabla\Psi\ dV green_2 \tag{4}$$

Subtracting equation **??** and **??** we get

$$\int_V \Phi\nabla^2\Psi - \Psi\nabla^2\Phi = \int_S (\Phi\nabla\Psi - \Psi\nabla\Phi)\cdot\mathbf{n}\ dS = \int_S \Phi\frac{d\Psi}{dn} - \Psi\frac{d\Phi}{dn}\ dS \tag{5}$$

This is known as Green's second identity.
$\nabla^2\Phi$ or $\nabla^2\Psi$ is the divergence of the electric fields, and since there are no current sources in the interior of the conductor,the volume integral is equal to zero (Maxwell's first law). This gives us the equation

$$\int_S \Phi\frac{d\Psi}{dn}\ dS = \int_S \Psi\frac{d\Phi}{dn}\ dSrec_2 \tag{6}$$

If we multiply by -1 and the conductivity, σ, on both sides, this is the current density on the surface of the conductor and is zero everywhere except at the electrodes. Therefore equation **??** becomes

$$I_\Psi\Phi(CC) = I_\Phi\Psi(PP) \tag{7}$$

where Ψ and Φ are the potential drops between the current-carrying and pickup electrodes, respectively. The transfer impedance is then

$$Z = \frac{\Phi(CC)}{I_\Phi} = \frac{\Psi(PP)}{I_\Psi} \tag{8}$$

which is the same as(**??**).

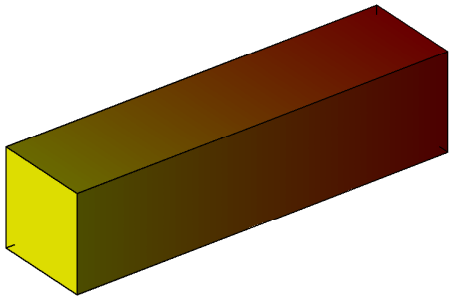


Figure 6.1: A homogenous, finite copper conductor with uniform current-density

6 Steady-state, analytical case in Matlab

We have several cases which have analytical solutions. Two of these can be seen in the figures 6.1 and 6.2.

6.1 The box-conductor

This is the simplest conductor imaginable, made of copper. We apply a fixed voltage to the the left side and ground the right side. Since the side walls are insulated, there will be a uniform current density along the entire conductor. We can easily calculate the impedance, or resistance since we assume no dispersive properties. The resistance, R, is $R = \rho \frac{L}{A}$, where L is the length of the box and A is the surface area with normal vector in the length direction. The resistivity of copper is tabulated as $1.67 \times 10^{-6}\Omega\ cm$. This gives us the resistance

$$R = 1.67 \times 10^{-6}\Omega\ cm \frac{4\ cm}{1\ cm^2} = 6.68 \times 10^{-6}\Omega$$

which furthermore gives us a current $I = \frac{V}{R} \approx 0.15 \times 10^6\ A$ As a final step (for comparison with the other programming environments), we will calculate the value of the current density $J = \frac{I}{A} = 0.15 \times 10^{10} A/m$. The potential is trivially $V_0(1 - x)$.

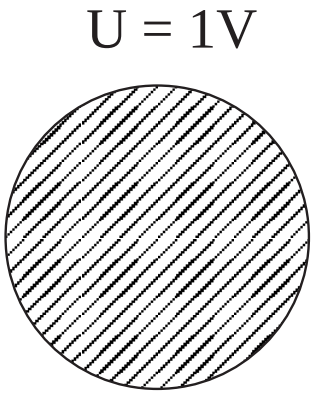


Figure 6.2: An infinite, homogenous copper-conductor with a spherical electrode in the center

6.2 A sphere conductor with central sphere-electrode

We imagine an conducting media with a spherical electrode in the middle. Furthermore, we again apply a fixed voltage to the electrode. This is the type of potential sent from a Greater Power, since there are no current wires or displacement currents going to and through the electrode. The outer conductor is grounded. We will quickly derive the potential Φ at a distance r from the center (outside the center electrode). We use the Laplace-equation from the previous section,

$$-\nabla \cdot (\rho \nabla \Phi) = 0$$

This equation has solutions of the form

$$\Phi(r) = Ar + B$$

The boundary conditions are as stated earlier, a potential V_0 at the center electrode, radius R_1. The outer shell is grounded, so that the potential at the radius R_2 zero. We substitute the variable r for r' so that r' = 0 at the boundary of the inner electrode.

$$\begin{aligned} r' &= R_1 - r \\ R_1' &= 0 \\ R_2' &= R_1 - R_2 \\ \Delta R &= R_2 - R_1 \end{aligned}$$

These conditions give the following solutions to the equation for the potential,

$$\begin{aligned}
\Phi(R_1' &= 0) = V_0 = A \cdot 0 + B = V_0 \\
\Phi(R_2' &= R_1 - R_2) = 0 = A \cdot R_2' + B = 0 \\
B &= V_0 \\
A &= -\frac{V_0}{R_2'} \\
&\Downarrow \\
\Phi(r') &= -\frac{V_0}{R_2'} r' + V_0 \\
&\Downarrow \\
\Phi(r) &= V_0 \frac{V_0}{\Delta R}(R_1 - r) + V_0
\end{aligned}$$

Part II
Programming

7 Introduction and presentation of the programming environments

There exists a plethora of programming environments, tools and graphical interfaces for simulating electro-magnetic systems. My predecessor Vegeir Knudsen has spent great time and effort distinguishing between the packages EMAS and Diffpack, where it eventually turned out that EMAS gave poor results for AC-signals [1].

In my thesis, I started where Vegeir left, using a test case to compare Diffpack and Femlab solutions. I then tested Femlab on some analytical test cases, with analytical solutions computed in Matlab. I have given a short presentation of the different programming environments below.

7.1 Matlab®

Matlab from Mathworks, short for *Matrix Laboratory*, is probably the worlds biggest commercial software package for numeric calculation. It is, as the name implies, based entirely on matrices, and all data must be entered in some kind of matrix. In the programming syntax, or scripting, it is very similar to most object-oriented languages. One defines objects of types, with inheritance and other aspects of classes. One can also define functions, or use one of the thousands of built-in functions. For this thesis, Matlab v6.5 has been used.

7.2 Diffpack©

Diffpack from inuTech GmbH is a framework for solving partial differential equations for the C++ programming language. It is designed to solve most classes of differential equations, and includes different tools from the finite difference and finite element methods. It also includes tools for graphical output and GUIs, but a thorough knowledge of C++ is required to utilize its functions.

7.3 Femlab©

Femlab by Comsol is both a function- and class library as well as a graphical user interface. Again from the name, Femlab is a tool for running simulations using the finite element method, which we will return to in a minute. It is very important to point out, that when it comes to all types of graphical user interfaces of the type 'point-and-click', as well as using any device at all, that without knowing what the button you press actually does, you will probably run head first into a wall at some point.

We started out using Femlab 2.1. Halfway through the thesis we switched to Femlab 3.0, which is a remarkably improved version when it comes to visualization speed. This is because this version does not rely on calling Matlab-functions, but is built up from the bottom using java-code. There is still cross-compatibility with Matlab-scripts, which is an important tool in making varying i.e. physical parameters without waiting hours at a time for the intermediate results. At the very end of the year v3.1 arrived, which primarily is upgraded to remove many of the scale-issues we will discuss later.

When it comes to Femlab, the computation time is of course mostly dependent on the number of elements in the simulation, or the resolution if you will. It is also dependent on the type of solver. In general, we use the UMFPACK (Unsymmetric MultiFrontal method Pack) for 2D-cases. This is a general Lower-Upper matrix reduction algorithm for non-symmetric, sparse matrices. Since our matrix is neither non-symmetric nor sparse, we are using too much computing power for to simple a case. But in this treatise we will not use an excessive amount of computing power on 2D-cases. In any case the solve-time is proportional with the number of elements.

It is in the 3D-case we need to pay attention to what kind of algorithms we are using. Our general workhorse will be a Conjugate Gradient method with a algebraic multigrid pre-conditioner. For our test-cases the solve time is also linear with the number of elements.

7.4 The Machine Environment

The simulations were done on a Pentium® Celereon© 1.30 Ghz with 1 GB RAM.

8 Finite Element-Method

A rigorous description of the finite element-method is given in Langtangen's book[3], and the material in this section is taken from there. The Finite Element method is a way of solving partial, differential equation (PDE) based

on approximating a function by a sum of *basis functions*, for example a sum of sinus functions. That is, we approximate a given solution, u, by a approximated $\hat{u}$ which is the sum of N functions.

$$u \approx \hat{u} = \sum_{j=1}^{N} u_j N(x)_j \tag{9}$$

where u_j are constants.

We furthermore divide our domain into a set of elements, where the basis function is a polynomial over the the elements.

The *gridsize* is therefore in this case the same as the number of elements in the model.

8.1 An Example With A 1-Dimensional Potential

We will use the following example. We have a 1-dimensional conductor, with conductivity 1 for easy computation. As boundary conditions we have a potential of 1 at $x = 0$, and an electric field equal to 1 at $x = 1$. In other words, the PDE to solve is the following:

$$\begin{aligned} x &\in [0..1] \\ -\nabla^2\Phi &= 0 \\ \Phi(0) &= 1 \\ E(1) &= -\frac{\mathrm{d}V(1)}{\mathrm{d}x} = 1 \end{aligned}$$

In the method called *Galerkin's Method*, it is quite common to use basis functions with the following properties:

- N_i is a polynomial over each element, uniquely determined by its values at the nodes in the element.
- $N_i(x^{[j]}) = \delta_{ij}$

δ is the Kronecker-delta, defined as

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

The last property ensures that the approximated solution, $\hat{u}$, has the values u_i at the nodes. A sketch of 1st degree basis functions is included in figure 8.1.

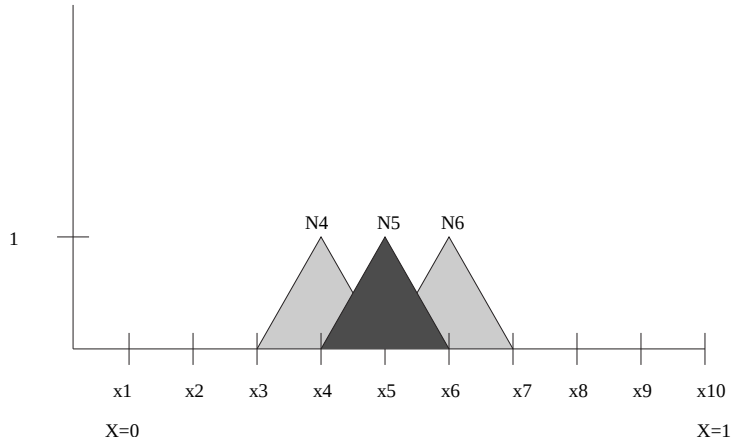


Figure 8.1: The shape of 1st degree polynomials over the elements. One can see that the polynomials are equal to one at their respective node, and zero at all others.

The great strength of the method is that one is quite free to choose the degree of the polynomials as well as the shape of the elements. This gives great flexibility towards the complexity of the problem. In our case, for instance, we can easily include 2 dimensions and non-trivial boundary conditions.

In our problem, we will approximate the potential Φ by

$$\Phi \approx \hat{u} = \sum_{j=1}^{m} N_j u_j \tag{10}$$

If we insert the approximated solution, $\hat{u}$ into the PDE, we get a residual , R

$$R = \nabla(\rho\nabla(\hat{u})) \approx -\nabla(\rho\nabla\Phi) = 0$$

We want the error in the PDE to be as small as possible, that is we want to minimize R. The goal is that by minimizing the error in the PDE, we will also minimize the error in the solution. In Galerkin's method, which is a *weighted residual method*, we integrate R multiplied by the weighted basis functions N_i over the domain Ω.

$$\rho \int_{\Omega} N_i \nabla^2 \, \hat{u} \, \mathrm{d}\Omega = \int_{\Omega} \sum_{j=1}^{M} N_j \; N_i'' \; u_j \, \mathrm{d}\Omega = 0, \qquad i = 1, \ldots, M$$

assuming ρ constant. Using integration by parts, we can replace the integral

above with

$$\sum_{j=1}^{n} \int_{\Omega} N_i' N_j' u_j \, \mathrm{d}\Omega - \int_{\Gamma} N_i N_j' u_j \, \mathrm{d}\Gamma = 0, i = 1, \ldots M$$

We will replace the integral on the boundary with the fitting boundary conditions when we get that far, and disregard it for now since it acts only on at $x = 0$ and $x = 1$. The equation above constitutes a matrix equation

$$A\mathbf{u} = \mathbf{b} = \mathbf{0}$$

where the matrix elements are equal to the product of the derivatives of the basis functions.

$$A_{ij} = \int N_i' N_j' dx$$

Looking at figure 8.1, we see that the matrix elements will be non-zero only for the neighboring basis functions. The derivatives of the functions are

$$N_i' = \pm \frac{1}{\Delta x}$$

where Δx is element size. Therefore the elements are

$$\begin{aligned} A_{i,i-1} &= -\frac{1}{\Delta x} \\ A_{i,i} &= \frac{2}{\Delta x} \\ A_{i,i+1} &= -\frac{1}{\Delta x} \end{aligned}$$

For the boundary conditions, we see that replacing the first and the last column elements, $A_{1,j}$ and $A_{10,j}$, with 0, A_{11} with one, and setting the first entry on the right hand side, $\mathbf{b}_1$ equal to 1, fulfills the first boundary condition, also called an essential or Neumann boundary condition. For the second, we remember that our original function included a term for the boundaries. The boundary condition can be regarded as

$$u(1)' \approx \hat{(u)}'(1) = u_{10} N_{10}' = E(1) = -1$$

so that we can replace the last vector element on the right and side with N_{10}, which is exactly equal to one. This is called a natural or Dirichlet boundary condition.
We then have the following matrix equation

$$\begin{pmatrix} \Delta x & 0 & 0 & \cdots & \cdots & 0 \\ -1 & 2 & -1 & \ddots & \ddots & \vdots \\ 0 & -1 & 2 & -1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & 0 & -1 & 2 & -1 \\ 0 & \cdots & \cdots & \cdots & -1 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ \vdots \\ \vdots \\ u_9 \\ u_{10} \end{pmatrix} = \Delta x \begin{pmatrix} 1 \\ 0 \\ \vdots \\ \vdots \\ \vdots \\ 0 \\ -1 \end{pmatrix}$$

This equation system can be easily solved by hand, but to save ourselves some trouble we used Matlab.

Doing this, we get exactly what we want, a function u that linearly declines to zero. The equation can be shown to give the exact solution to this particular problem, but only at the node-points.

8.2 Boundary Conditions

There are several different boundary conditions we can apply do boundaries in our models. Two have already been mentioned, the fixed potential and electric field conditions. Between different materials we will generally be using the 'continuous' condition. This means that the the electric field, will be the same on either side of the boundary, or to put it in more mathematical terms:

$$\frac{\mathrm{d}\Psi(x+\delta)}{\mathrm{d}x} = \frac{\mathrm{d}\Psi(x-\delta)}{\mathrm{d}x}$$

Later on we will be using the thin conducting layer condition. This is modelled as a thin conducting sheet, connected to a potential V_0. The equation for this boundary-condition is:

$$\mathbf{n} \cdot \mathbf{J} = \sigma \frac{V - V_0}{d}$$

8.3 Extending to More Complicated Problems

Thankfully we will handle more complicated problems in this thesis than only one-dimensional problems, the first extension being 2- and 3-dimensional cases. In this case, we will have a matrix or *mesh* for which to solve the problem. If we had a highly regular geometry, like a box, we could simply extend the node-points in the other direction, so that we could express the potential as

$$\Phi(x, y) \approx \hat{u} = \sum_{j=1}^{m} \sum_{k=1}^{n} N_{jk} u_{jk}$$

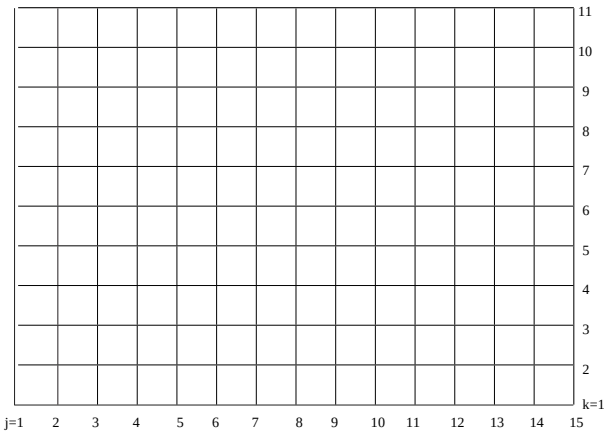


Figure 8.2: A two-dimensional mesh on a box-object

where j could be the index in the x-direction and k the index in the y-direction, as modelled in figure 8.2

One can see how placing these square elements becomes difficult with increasingly complex models. A much more flexible way to place node-points, and the method which is commonly used in the finite-element field, is to use triangle elements instead, as depicted in figure 8.3

These elements are more flexible in handling corners and points on a geometry because the triangle sides can be stretched without altering the properties of the basis-functions.

We can further extend to the 3-dimensional case using tetrahedrons.

We will usually let Diffpack or Femlab handle all the details of the integration, the number of basis-functions, type of element and type of numeric integration over the element. But we can also take control over these factors, for a tighter grip on the solution and error sources.

9 Finite-element treatment in Diffpack

How we do things in Diffpack is naturally to solve a set of differential equations, in this case Laplace's equation from chapter one. The Diffpack environment provides us with an easy setup for this, and the problem can be readily solved as a special case of Poisson's equation which there is ample

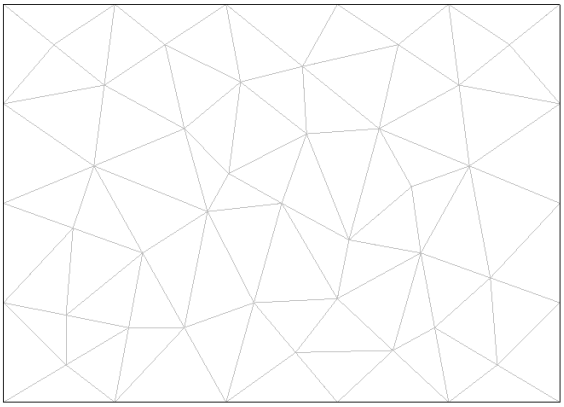


Figure 8.3: A triangle-mesh

documentation of[3]. In anticipation of more complicated geometries, the *finite element model* seems to be the most useful tool for solving our problem. The reason for this is that once the basis functions and element types have been chosen, adapting them to a geometric model is reduced to just being able to fit the elements on the model.

The model we are going to solve with Diffpack is the following; A 2-dimensional rectangle with resistivity equal to unity, with four electrodes attached on top. The 3-dimensional equivalent to this is a box with (infinite) cylinder electrodes. A sketch of the geometry is included in 9.1.

Boundary Conditions and Materials

One of the most important features of Diffpack is the ease with which it handles boundary conditions. While not simple at all when it comes to the analytical case, coding an insulating wall or a materials with various electrical properties becomes an easy task. It is simpley a matter of setting switches on and off. For our problem, we have the following boundary conditions.

We will assume that our medium is totally isolated except for at the electrodes. This implies that there is no current in or out at the non-electrode boundary, or that $\frac{\partial V}{\partial n} = 0$, known as the homogenous Neumann condition. At the electrodes we will apply two external voltages, V_1 on the outer electrodes and V_2 on the two inner electrodes. This gives us four different potentials,

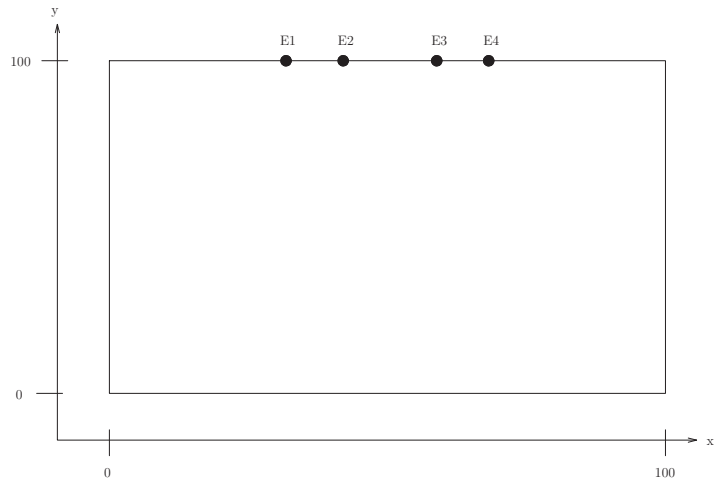


Figure 9.1: Model of the test case for comparison between Diffpack and Femlab

$+V_1$, $+V_2$,$-V_2$ and $-V_1$. These imply four different Dirichlet conditions.

Mathematical Problem

$$\nabla^2\Phi(x,y) = 0 \tag{11}$$
$$\frac{\partial V}{\partial n}(x,y) = 0, \quad x \epsilon \Omega_E \tag{12}$$
$$\Phi(x,y) = V_1, \quad x \epsilon \Omega_{E_1} \tag{13}$$
$$\Phi(x,y) = V_2, \quad x \epsilon \Omega_{E_2} \tag{14}$$
$$\Phi(x,y) = -V_2, \quad x \epsilon \Omega_{E_3} \tag{15}$$
$$\Phi(x,y) = -V_1, \quad x \epsilon \Omega_{E_4} \tag{16}$$

The electrode geometry is thus considered to be something akin to band-electrodes. Initially hemispherical boundaries were considered, but this was regarded as somewhat of an obstacle in Diffpack, and beside the point. It does, however, pose a big problem since this case cannot be considered analytical anymore. But we can, and will, solve it in Femlab for a comparison between the two.

The main reason that we code in only two dimensions is that it saves a lot of computation time. Also, if we ran into trouble in the simulation debugging would be a slightly more hazardous course in three dimensions, as the equations need a few more terms and the geometry starts getting complicated. We will in any case return to the 3D case in a little while.

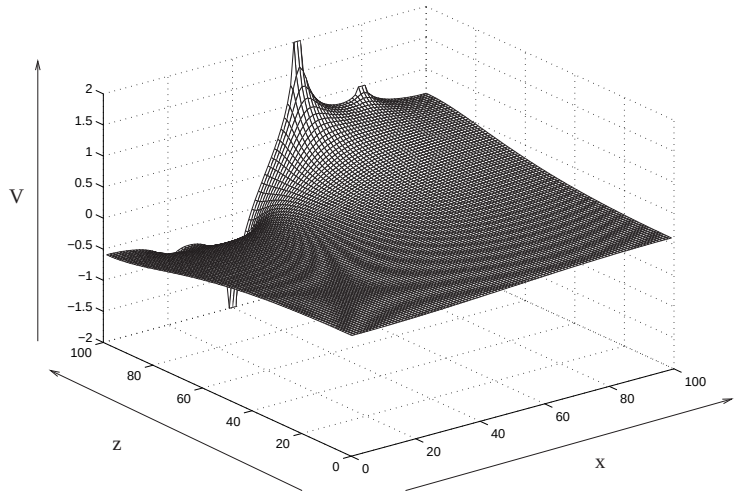


Figure 9.2: Potential-plot of the simulated DC-case from Diffpack. The potential is plotted along the V-axis

The boundary conditions are easiest included by an input script. The script, as well as the entire program, is included in the appendix. The result of the simulation is plotted in 9.2

10 Comparing Diffpack and Femlab

The same problem has been solved with Femlab. A series of operations have been made in the Femlab-GUI. These can be seen in the Matlab M-file called "Femlab-script for DC-case" (included in the appendix). These commands will naturally seem a bit cryptic to one not familiar with Femlab, Matlab and Finite-element syntax. This is of course why it is more convenient to work with the GUI.

So how does this one compare with Diffpack? The answer is; not to bad, and not to well. The results are too similar to tell apart next to each other, but if we make a difference plot as in figure 10.1, the difference becomes quite clear.

The difference here is suspected to arise from the different element types being used, and from the interpolation process. It would require quite some work to solve the problem with the exact same conditions in both programs, and this is not the problem at hand.

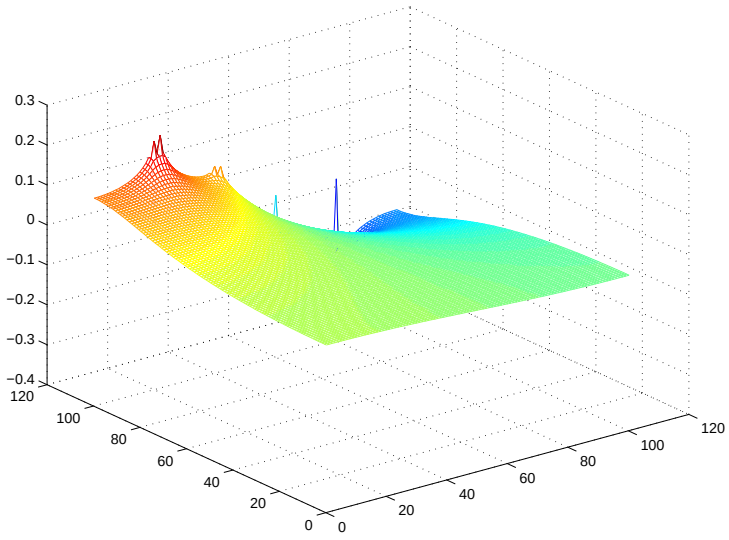


Figure 10.1: Plot of the difference in simulated potential in Diffpack and Femlab

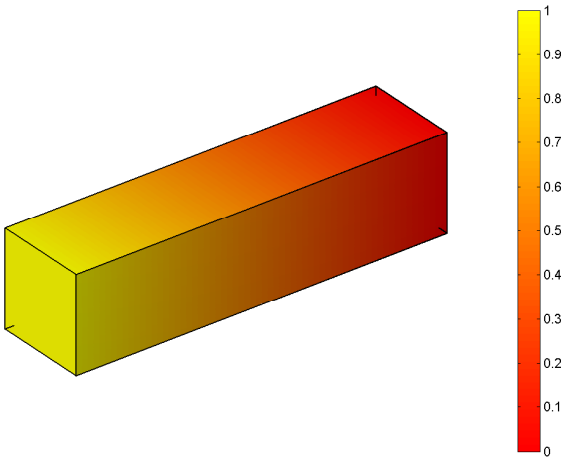


Figure 11.1: Plot of the voltage in a box conductor, one side clamped at 1 V, the other side grounded

11 Comparing Analytical Cases with Femlab

The next thing on the agenda is to compare the analytical cases with the simulated. We will build models to compare with the analytical models solved in the previous part.

11.1 The Box revisited

We return to the box from the previous section. Simulating a box like this is trivial in Femlab, so we will not go further into the design process. A plot of the simulated potential in the box is shown in 11.1. The conductivity of the model does not matter as long as we are just looking at the potential, but comes into play when we look at current and current density. As expected, the current and current density are constant over the entire domain.

In Femlab we can also do a boundary integration across the grounded wall (or any surface we choose, for that matter), and doing this we get exactly the same current that we arrived at in the analytical treatment, $1.4975 \cdot 10^9 A/m^2$.

We can enter an analytical expression of the voltage as a scalar expression in Femlab, $V = 1 - x$. We can also enter in an expression for the error, and the error squared. A plot of the error squared for a 2-D and 3-D model is included in the figures 11.2 and 11.3 for 2-D, and 11.4 for 3-D. A plot of the error as a function of the grid-size is included in 11.5, which is a the number

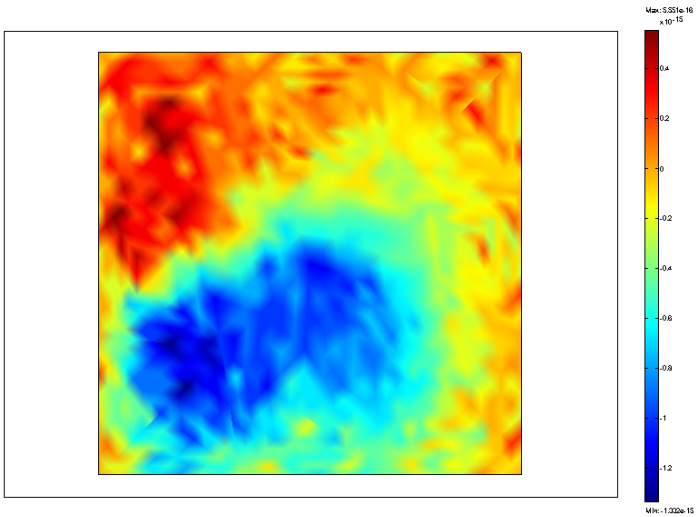


Figure 11.2: The error squared for a coarse grid on a 2-D model

of elements in the model, a measure of resolution.

As we can see, there is not much to be gained by increasing the grid-size. This could be expected, since the error is already close to the machine number (the smallest number possible to handle in computations). In fact, increasing the resolution further can be expected to increase the error, since small error-fluctuations could lead to large fluctuations that remain uncompensated by boundary conditions very many elements away. And this is disregarding the computational effort and time, which increases roughly by N, where N is the number of elements.

11.2 Infinite conductor with spherical electrode

This is a more serious example since curved edges and surfaces come into play. We have a spherical electrode surrounded by vacuum and want to calculate the potential at a distance r. At an outer boundary (meant to be at infinity) the potential is set to ground.

There is a problem regarding comparison with the analytical case, since we now have a finite conductive media. We will show that a measure of error,

$$\epsilon = \int_V V_{Anal} - V_{Simul} \; dv,$$

decreases with increasing size.

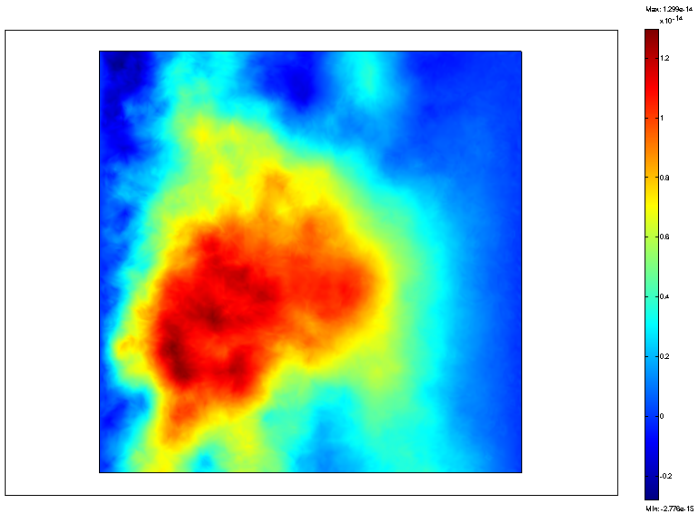


Figure 11.3: The error squared for a finer grid on a 2-D model

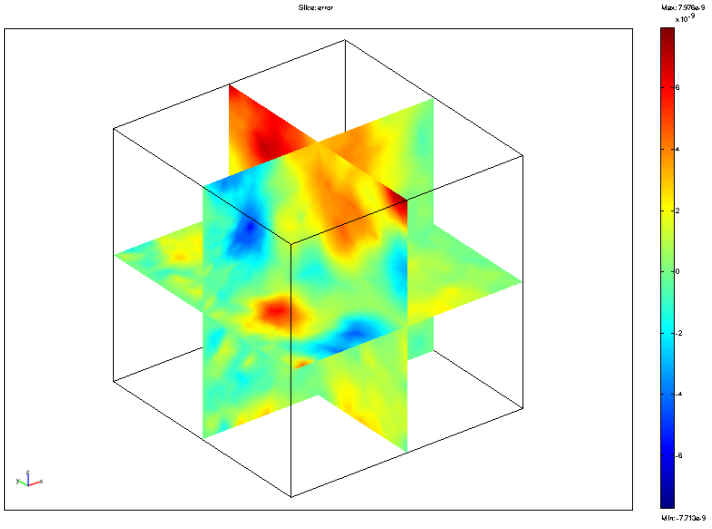


Figure 11.4: The error squared on a 3-D model

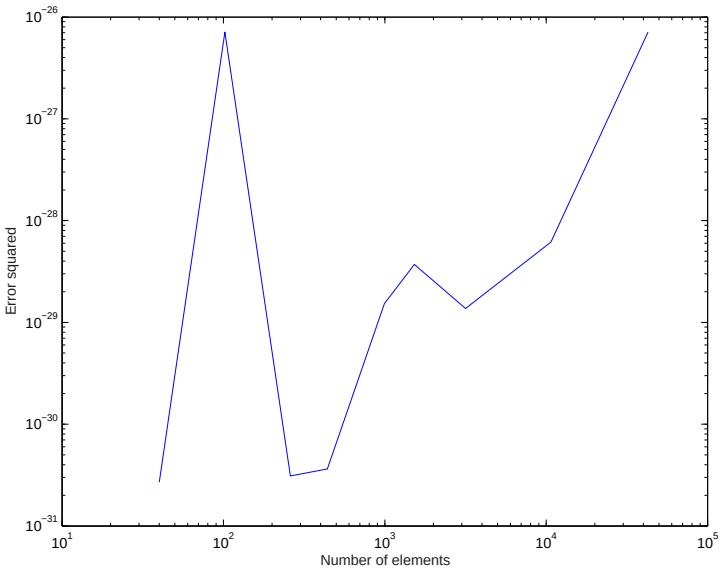


Figure 11.5: A log-log plot of the error squared as a function of the grid-size on a 2-D model

The analytical case was solved in the theory chapter. A Matlab script for the electric potential computation was produced, which closely resembles the script used for comparison between Diffpack and Femlab in the previous section. The difference is that the electrode is in the center of the geometry, and that there is only one. In figure 11.6 the analytical potential is plotted. The solution found in Femlab is plotted in figure11.7

They seem remarkably similar, one has to plot the error to really see where the difference lies. The error is plotted in figures 11.8 and 11.9. As seen from the figures, and as expected, the error decreases rapidly from 100% at r = 1m for a box with dimension 10, to 25% for a box with dimension 50. We can hope that this error goes to zero as the dimension goes to infinity. One does, however, also see in the graph the price paid for increased accuracy in the model. Solving for bigger and bigger models forces us to increase element size for the sake of computational time.

11.3 Finite conductor with spherical electrode

However interesting a comparison with (semi)infinite cases is, we are more interested in cases that we can both simulate and find analytical expressions for. This one is solved in the previous chapter, and is of course also quite trivial. A plot of the error for normal grid-size, and a plot of the square of

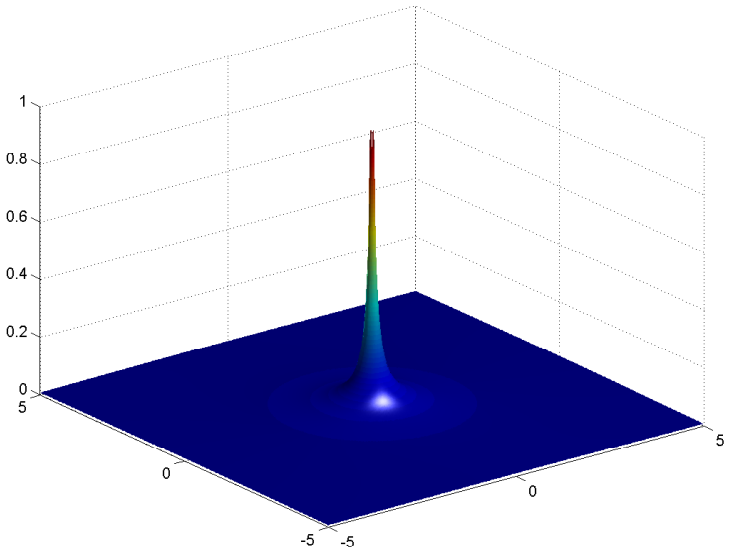


Figure 11.6: The potential of the analytical solution to an spherical electrode and an infinite medium

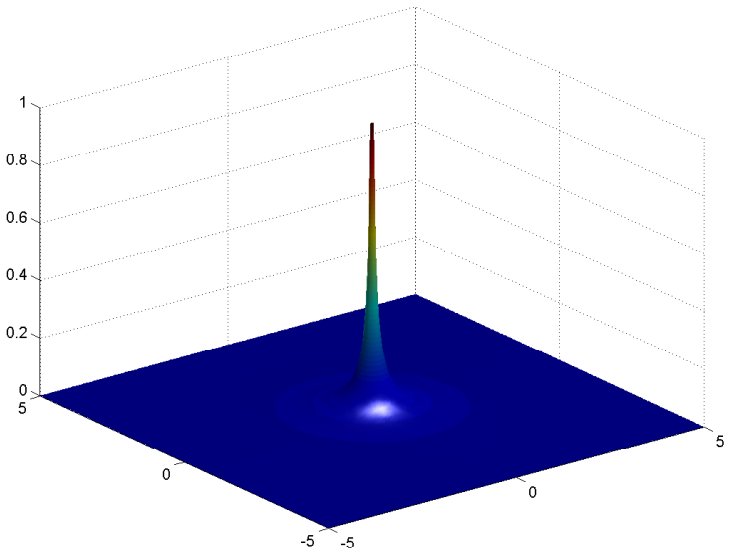


Figure 11.7: The potential found in Femlab, with the size of the model equal to 10x10

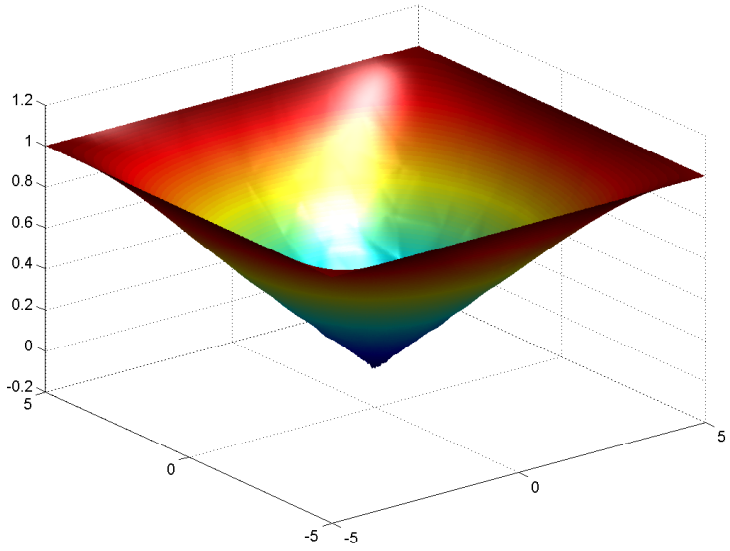


Figure 11.8: The error between the analytical and simulated potential, for a size 10x10 model

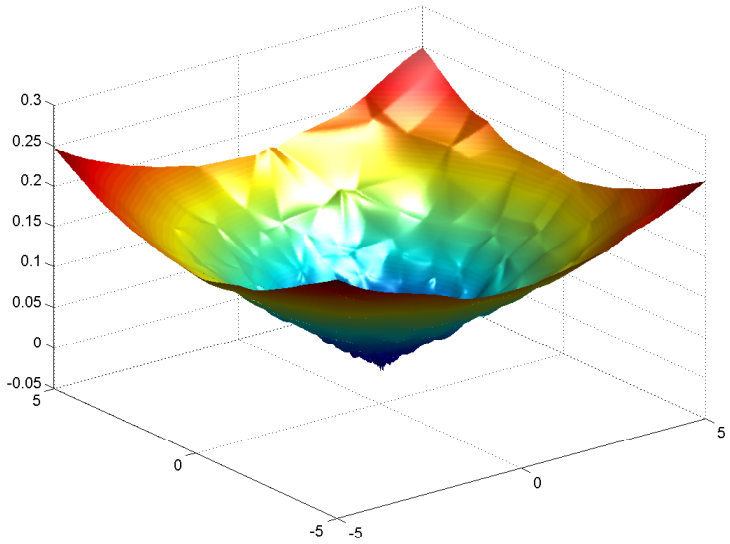


Figure 11.9: The error between the analytical and simulated potential, for a size 50x50 model

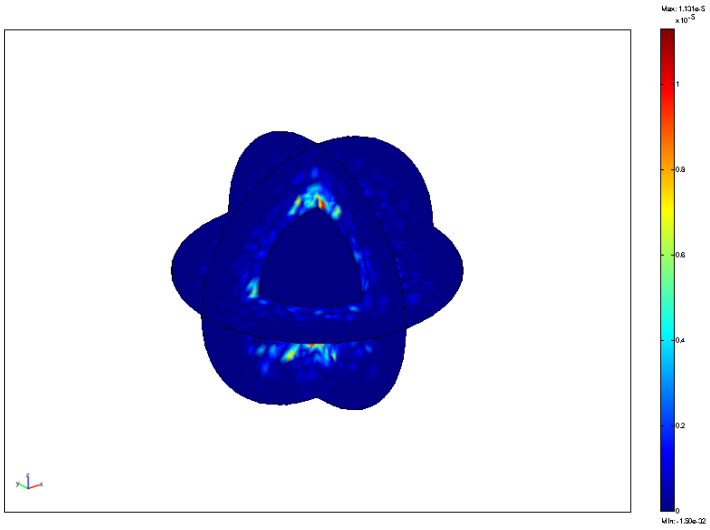


Figure 11.10: A plot of the error in a 3-D spherical conductor

the error vs. the grid size is included below in 11.10 and 11.11.

12 Results

It would seem that there is good reason to believe that Femlab will give accurate results for the problems we are going to be investigating in this treatise. For elementary problems like those in the previous section it gives errors on the scale of the machine number.

12.1 Detail Level and Computation Time

The fact that increased resolution might not necessarily give a more accurate solution is not the only reason to keep things simple; In the world of simulation increased detail also means increased computation time, and in some cases, our program might not even be able to handle the dimension of the problem.

By my experience in Femlab, having more than 150 000 elements in a model has usually meant that the simulation is unstable regarding memory, i.e. the solver-routine might break with an error containing an "out of memory" message.

The computation time relative to the number of elements is generally dependent on the solver. An example of the solution time relative to the

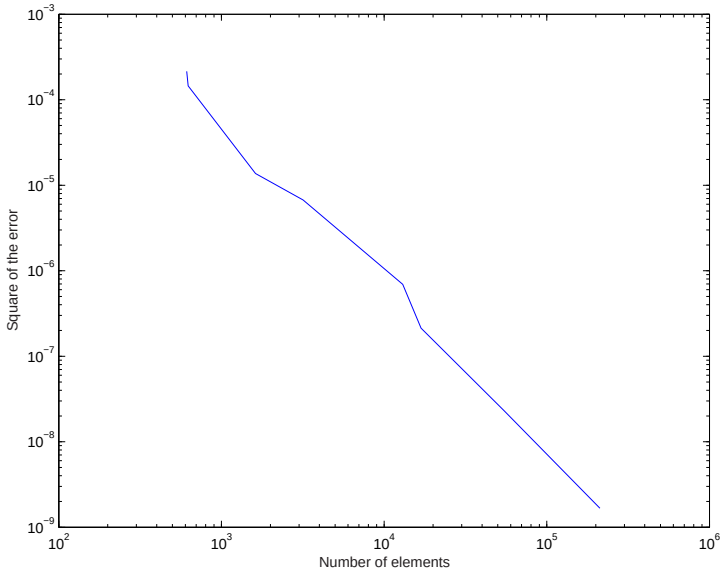


Figure 11.11: A plot of the square of the error as a function of the grid-size

number of elements for 2D-problems (with the UMFPACK-solver) is included in figure 12.1.

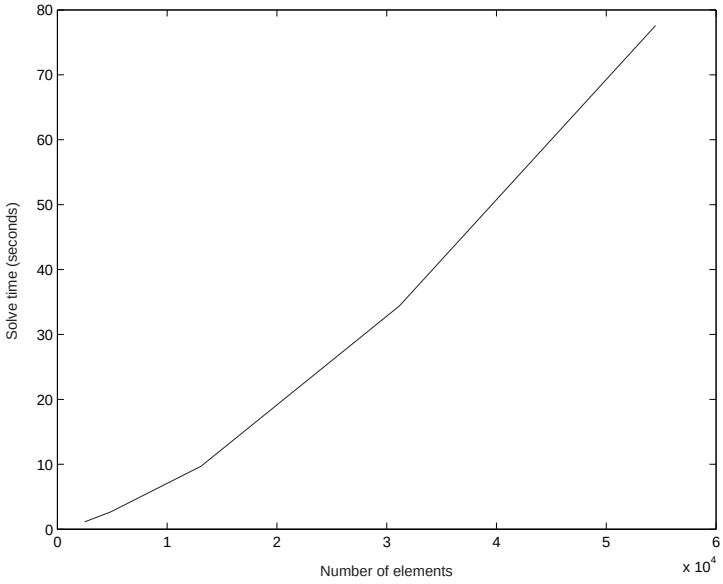


Figure 12.1: The solver-time vs the number of elements for a 2D-problem

Part III
Case Studies

13 Introduction to Case Studies

In this part we will analyze some simulations with different models. We will mainly look at the effect of the sensitivity distribution in varying the parameters discussed further below. All the simulations will be done in Femlab, with additional analysis with the use of Matlab.

- **Conductor geometry**

  We will investigate how size, shape and form influence sensitivity. It is generally known that at distances outside the *near-zone* (as stated in Basics[2]) the sensitivity can be approximated to zero. This means that changes (in resistivity) here will not give any change in the pickup signal.

- **Conductive isotropy**

  Most literature on bioimpedance presupposes isotropic conductivity. But in tissue this is the exception rather than the rule. We will analyze cases where the resistivity varies in two or three dimensions, parallel, orthogonal to linear electrode configurations. We will also investigate how isopotential lines can demonstrate anisotropy.

- **Electrode geometry and configuration**

  The bioimpedance field has a plethora of electrode types and electrode setups to choose from which are designed for different purposes. The main types to be discussed are the band-, spot-, needle- and circular electrodes.

- **Complex materials** Most biological tissue has an imaginary component of the impedance. A case with a sinusoidal current is applied will be simulated.

- **A Cylinder** A special test case where a cylinder conductor with two circular electrodes on the surface and two on the endcaps is simulated. A comparison with a axis-symmetrical model is done.

14 Scale and Dimension

In these investigations, we were not expecting to get results comparable with experimental measures. In most of the cases, all dimensions, length, width and height, extension and variables will be given without scale or unit. This is to save us the work of taking proper care of the units, since these are given to compare with the constants of the SI-system.

Since the conductivity σ and permittivity ϵ are typically inserted with tabulated values of copper and vacuum, respectively, the resulting values of the potential might be correct, but for the scale given. In retrospect, further attention should have been given to use the same scale on all models, to make them comparable.

15 Conductor Geometry

In most analytical cases, a lot of ideal conditions are pre-supposed, for instance superconductive electrodes, homogeneous, infinite materials and so forth. One of the effects supposed to have a very strong influence is the case where current has limited space to flow. This is an important case for another reason; In our later simulations, we will always have to deal with finite sizes, since we need a finite area to assign our partial differential equations. It is therefore important to acquire some feeling of how much this affects the impedance.

We will attack the problem of geometry from three different approaches

1. Different sizes in the direction parallel with the electrode array
   The length of the conductor is regarded as the extension in the direction parallel with a line through the electrodes, regardless of what kind electrodes we are dealing with. This is best demonstrated by an illustration, as in figure 15.1. We will consider a model infinite in the width-direction, so we can model it in two dimensions.

2. Different depths, with electrodes on "top"

3. Different widths, where width is the direction normal to the length of the conductor

In all of the cases we will see the effect on transfer impedance, and try to explain some of them by the sensitivity distribution.

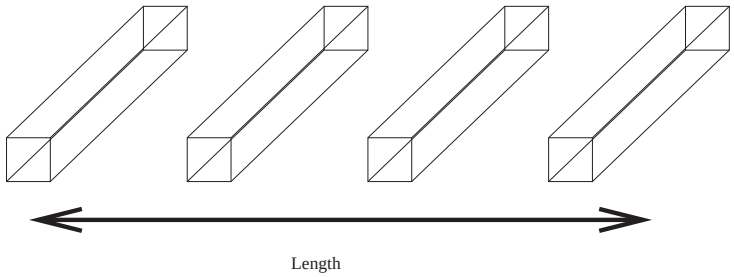


Figure 15.1: A model to describe how we define the length of a conductor, which is parallel to a line which runs through a electrode array

15.1 Conductor Lengths

We return to the familiar case of a box conductor with semicircular electrodes. In this first case we will disregard conductor width, and only examine the case in two dimensions. The electrodes thus become half-cylinders. The electrodes are 20 apart, and the radius is 8. The height of the model is 100. For the mass-production of these values and images a Matlab script was made which is included in the appendix. An illustration of the model is included in figure 15.2.

The outer electrodes are considered current-carrying, while the inner are pickup-electrodes, over which we integrate the potential to get an average value. This means that that the potential is evaluated at each node-point included on the boundary of the electrode, multiplied by the area of the element.

Boundary Conditions

The entire surface of the conductor is considered insulating, while the boundary between the electrodes and the conductor is continuous. The materials are chosen so that the electrodes have the conductivity of copper $(7 \cdot 10^7)$, while the conductor has conductivity 1. The current-density through the electrode tops was set to 1, downwards and upwards through the leftmost and rightmost electrode, respectively.

Results and Reciprocity

The impedance vs the length is plotted in figure 15.3. Firstly, one can see that the impedance is reduced when one increases the length of the conductor.Also, as expected, the impedance goes asymptotically towards a plateau.

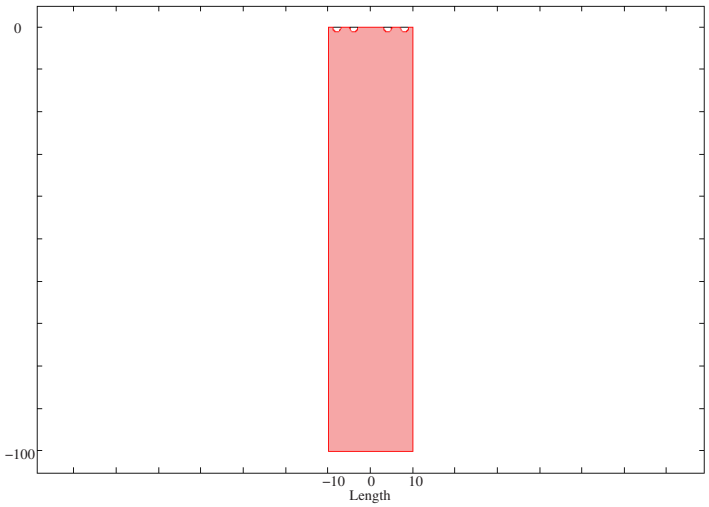


Figure 15.2: A model used for calculating the impedance vs the length. Notice the half-cylinder electrodes at the top of the model.

This is expected, as according to theory changes in local resistivity further out than the near-zone, should not influence the measured impedance.

The current was switched and run through the center electrodes in stead, while listening on the outer electrodes to measure reciprocity, measuring the reciprocal error ϵ as

$$\epsilon = \frac{Z_1 - Z_2}{Z_1}$$

where Z_1 and Z_2 are the impedances measured as the primary electrode setup and switching the electrodes, respectively. This produced an error approximately equal to the least machine number (10^{-14}) except for the value at a length equal to 70, for which a spike of 10^{-7} was found. No explanation for this spike has been found.
Some tests of impedance vs length were made on a 3-dimensional, anisotropic simulation. These can be reviewed in the next section on Anisotropy.

15.2 Effect of Finite vs Infinite Conductor Sizes

We can compare the sensitivity distributions of a conductor of finite and infinite length. A plot of the two is included below in figure 15.4 and 15.5. As we can see there is not a very significant difference in the sensitivity distribution in the two models. There is, however, a significant difference in the measured impedance in the two cases. In the infinite case, the impedance

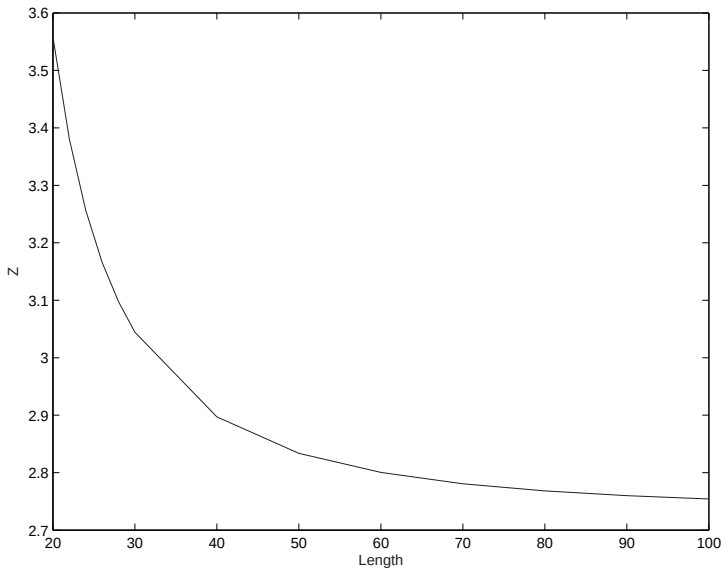


Figure 15.3: Plot of the transfer impedance in a homogeneous conductor vs the length of the conductor

is measured to be approximately half that of the finite model. The potential difference is lowered in the infinite case, possibly because the equipotential lines lie more "loosely" packed, so that the pickup electrodes lie closer to higher equipotential-lines. In figures 15.6 and 15.7 a plot of the equipotential lines of the two cases are compared.

15.3 Conductor Width

Now we turn our attention to the case where the width of the conductor varies. The width is defined as the direction normal to a line through the electrodes. In this model we have used electrodes shaped like cubes to have a look at the current density at the sharp corners. A plot of the model is included in figure 15.8.

Boundary Conditions

The boundary conditions are virtually the same as for the previous case-study, conductor length. In this case we have applied a symmetry-condition since the model is symmetric along the y=0-axis. The boundary condition thus becomes $\mathbf{n} \cdot \mathbf{J} = 0$, since there will be no current passing through plane. We can then remove half of the model, half the number of elements and 7/8

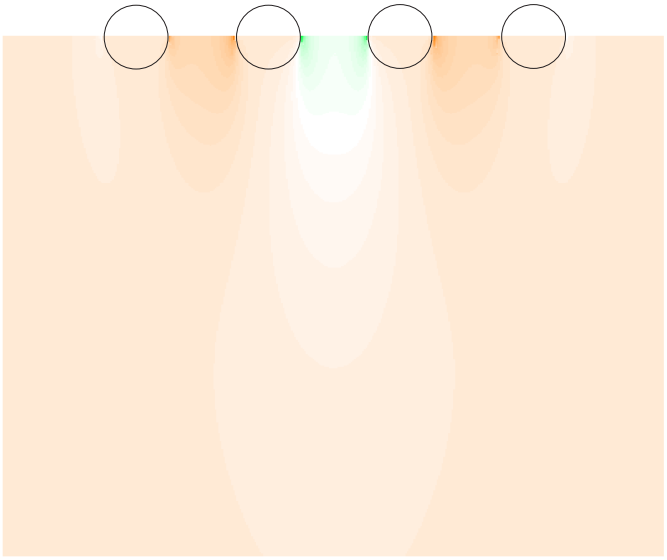


Figure 15.4: A plot of the sensitivity distribution on the semi-infinite medium. Green signifies positive sensitivity, and orange is negative

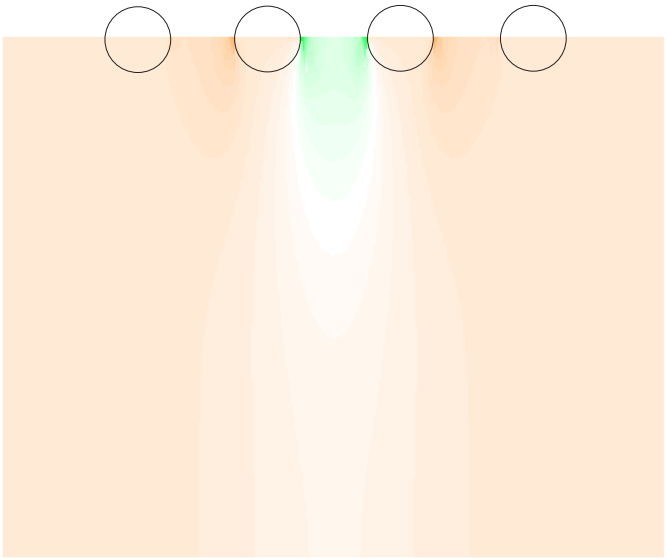


Figure 15.5: The finite medium

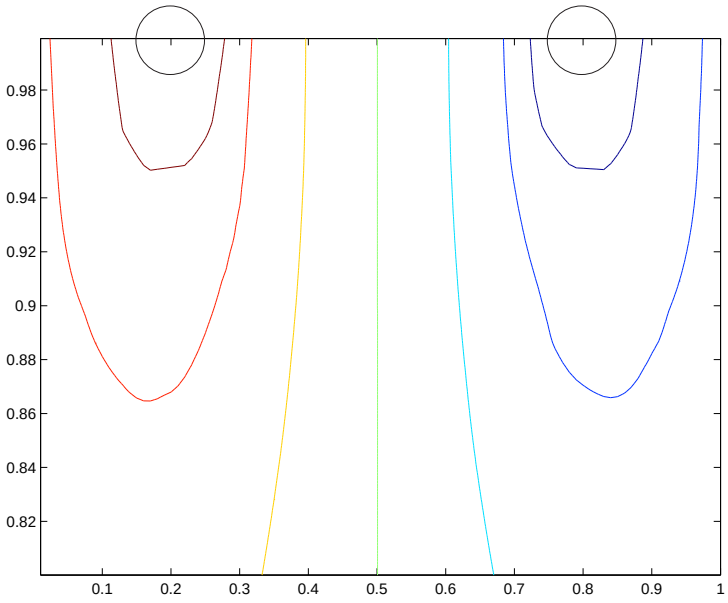


Figure 15.6: A contour plot of the potential in an infinite medium. The walls of the figure are just the frame of the plot

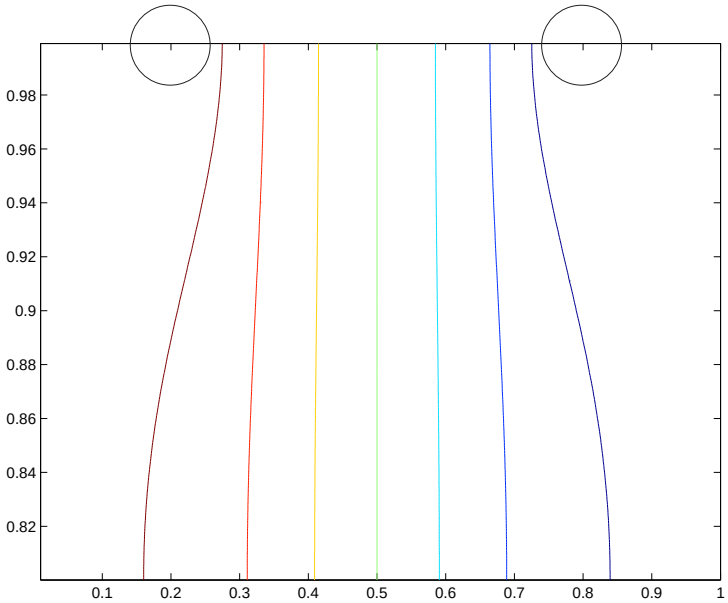


Figure 15.7: The finite medium. Notice that in this case the walls of the figure are the *actual* walls of the model

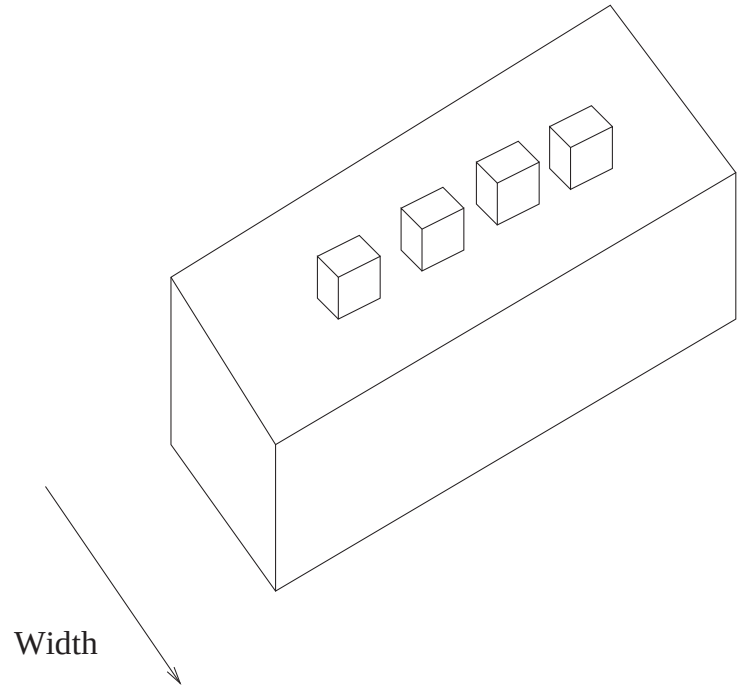


Figure 15.8: The model used for simulating the effects of the conductor width

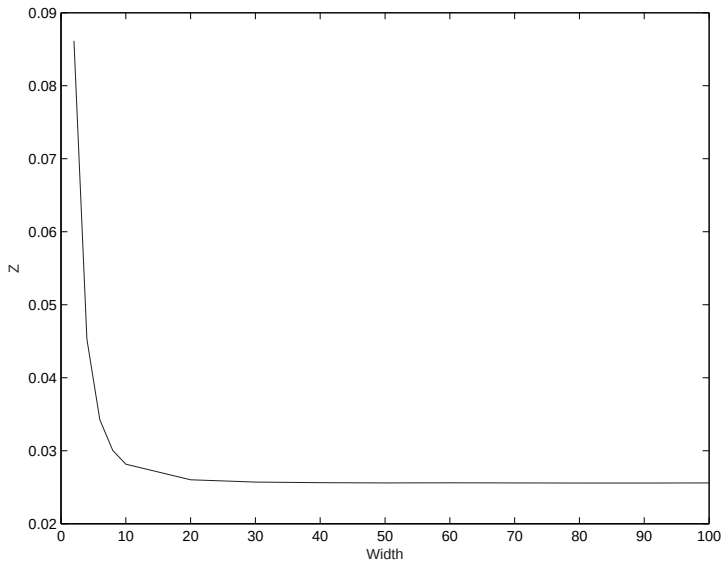


Figure 15.9: The impedance vs the width of a conductor

of the computation time.

Results and Reciprocity

A plot of how the impedance varies with the width is included in figure 15.9.

Here we can see how the impedance drops quickly to a plateau. We can intuitively presume that as the sample widens, the current density and the electric field decreases between the pickup-electrodes. Thus the integral

$$\Delta V = -\int \mathbf{E}\ \mathrm{d}\mathbf{x}$$

decreases. Very quickly, however, the sensitivity to changes approaches zero, and changes in the width have no effect on the impedance.

The current-carrying and pickup-electrodes were switched to measure reciprocity, measuring the reciprocal error ϵ as

$$\epsilon = \frac{Z_1 - Z_2}{Z_1}$$

where Z_1 and Z_2 are the impedances measured as the primary electrode setup and switching the electrodes, respectively. This produced an ϵ of the order $10^{-}9$, which can be considered good enough for our simulation.

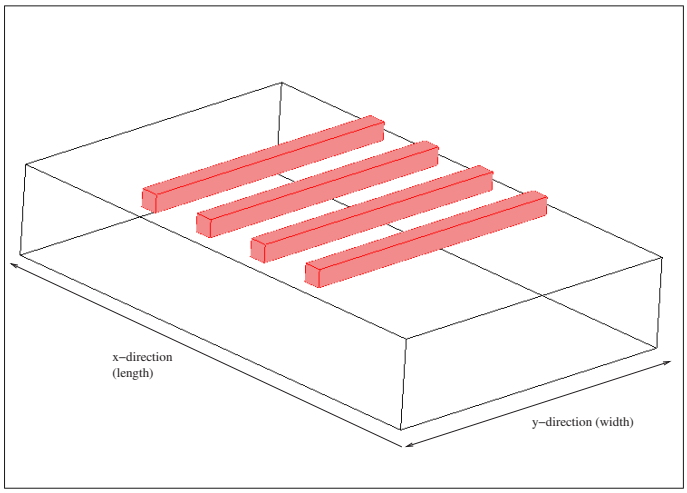


Figure 16.1: A plot of the model for impedance vs the electrode size in the high-conductivity direction

16 Anisotropy

Some simulations were made on different geometries, box conductors and spherical conductors.

16.1 The Box-Conductor, Revisited Again

A model was made with double conductivity in the x-direction. The geometry of the model was made to be comparable to a real measurement on a piece of Longissimus Dorsi from pig, done by Mr O.E.Rosseland. When flipping the electrodes 90 degrees, thus measuring in a direction with half the conductivity, the impedance was expectedly doubled.

Some tests where the size of the conductor was varied was done to try to explain anomalies in the real measurement. Here length is the x-direction with high conductivity. A figure of the model tested and the impedance vs. length plot is depicted in figure 16.1 and 16.2 below.

In the case where we flip the electrodes 90 degrees, the results are as expected from the previous section. Here the potential decreases, and the current-density rises. We thus have two effects which add to lowering the impedance. A simulation was done where we varied the length and width of the conductor, and kept the inter-electrode distance constant. A plot of the impedance as a function of length and width is included in figure 16.3.

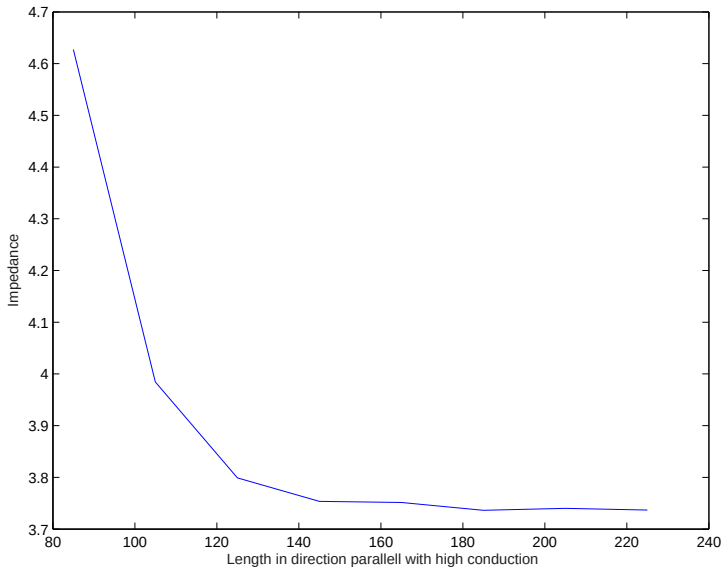


Figure 16.2: The impedance as a function of length

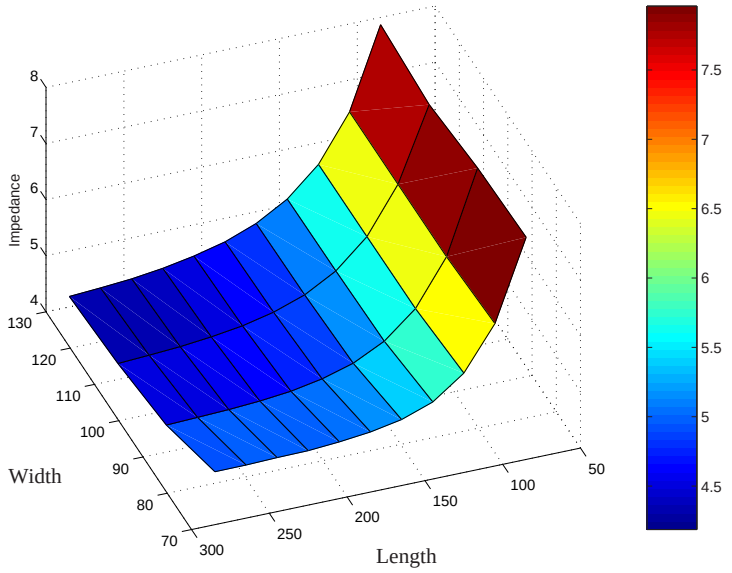


Figure 16.3: The impedance in a 3d-conductor with high conductivity in the x-direction, but with the electrodes flipped 90 degrees, vs length and width

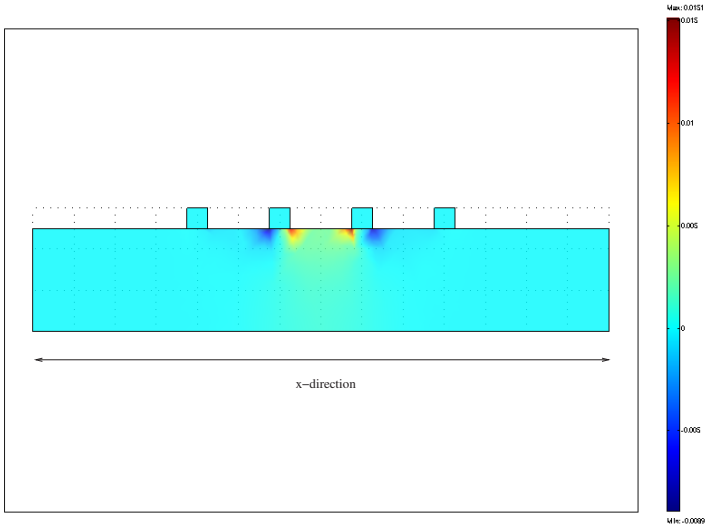


Figure 16.4: A sensitivity plot of the model with high conductivity in a direction parallel with the axis through the electrodes

We see that the effect of the width has a very small effect compared to that of the length. This could be interpreted as that the changes in the size of the conductor in the direction with high conductivity have a larger effect than change in the low-conductivity direction. But if we look at sensitivity plots in the parallel and transverse to the electrodes, we see that there is virtually no sensitivity outside the current electrodes in the width-direction, while there is a very small component in the length-direction. A plot of the two is included in figures 16.4 and 16.5.

This means that whatever we do outside the outer conductors in the width-direction is bound to have a very small effect since the setup is non-sensitive to changes here.
The impedance measured at the 90×140 base-area was calculated, with electrodes in both directions. As expected, the impedance in the parallel case is half that of the transverse case, since the conductivity is twice as high in that direction.

16.2 Concentric Electrodes

Doubling the resistivity in the main direction approximately doubles the impedance, and doubling it in the transversal direction does close to nothing at all when using stripe electrodes. What so if we use concentric electrodes? The anisotropy is defined as the fraction of conductivity in the x-direction,

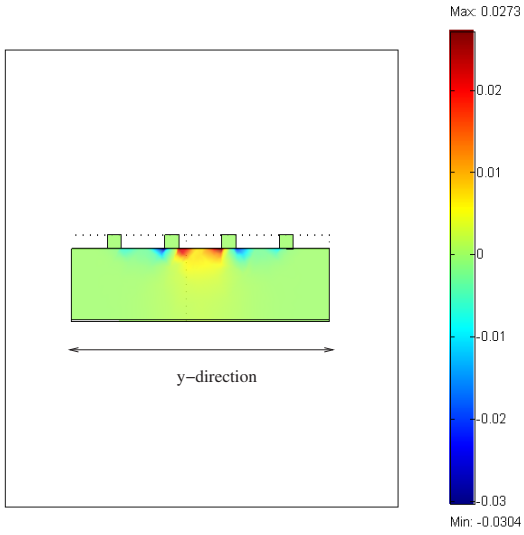


Figure 16.5: A sensitivity plot of the model with high conductivity in a direction normal to the axis through the electrodes

σ_x, over the conductivity in the y-direction, σ_y, so that we are looking for

$$Z(A) \text{ where } A = \frac{\sigma_x}{\sigma_y}$$

A plot of the model is included in figure 16.6. The width of the electrodes is 5, the height is 5, the inter-electrode interval is 5, and they lie on top of a box which is 200x200x200.

Boundary Conditions and Reciprocity

The outermost and innermost electrodes are used as current-carrying electrodes, where a current-density is computed according to the area of the particular electrode. From the theory chapter, we have

$$J = I/Area$$

We choose to have a unit current run from the innermost electrode to the outermost. Since the electrodes have different areas, we need to divide the current by this area to find the correct current density.

The electrodes have the conductivity of copper, the box has conductivity 1, but the conductivity in the x-direction is varied. The potential is measured on top of inner electrodes.

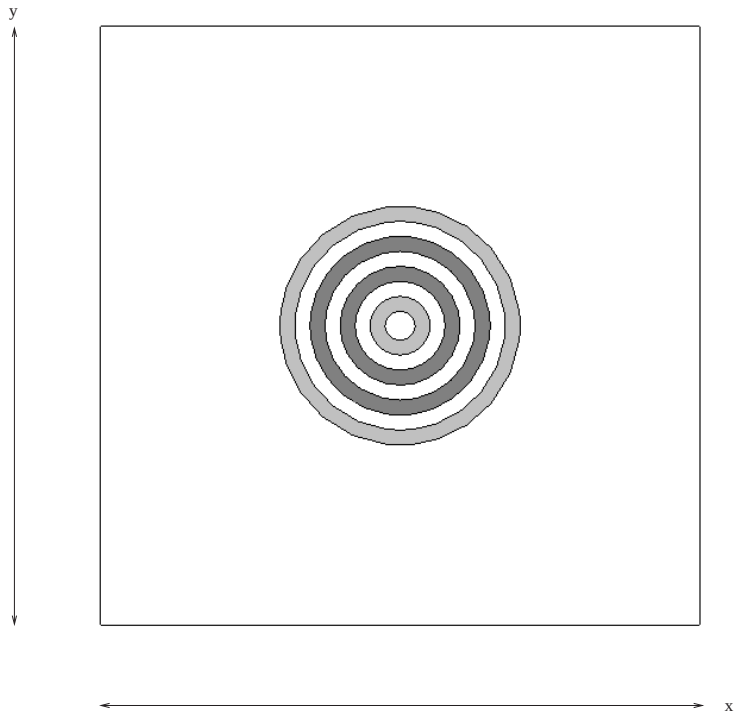


Figure 16.6: A plot of the model used for investigation of concentric electrodes, seen from above

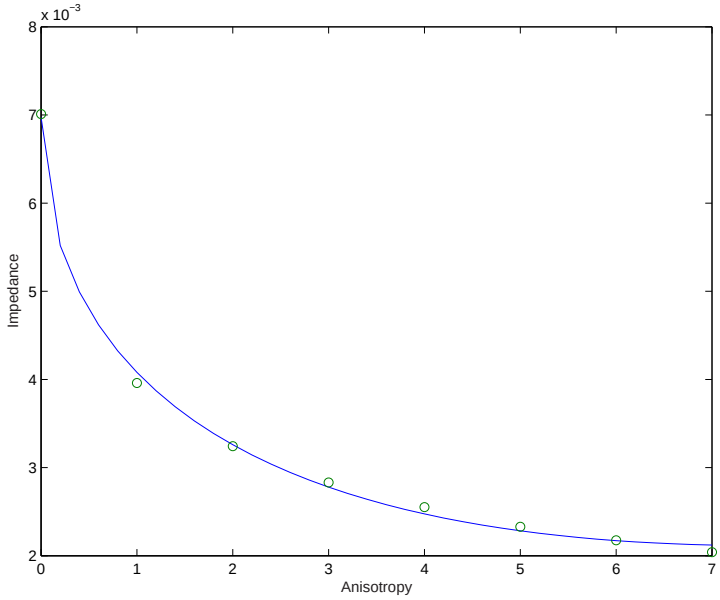


Figure 16.7: The impedance in a model with concentric electrodes vs the anisotropy of the medium

The current-carrying and pickup-electrodes were switched to measure reciprocity, measured as

$$Reciprocity = \frac{Z_1 - Z_2}{Z_1}$$

This gave results from 2-8 percent, with an important exception for A = 4:1, for which a reciprocity of 68 % was measured. This anomaly has as yet not found an explanation.

Results

A plot of the measured impedance vs the anisotropy is included in figure 16.7.

A fitting to a square-root function was made, and this can be seen overlapping the measuring points in the figure. This fitting gives a impedance that varies with the anisotropy as

$$Z(A) = 0.7 - 0.35A^2$$

The largest error is 1.2095e-004, which is pretty confident (the error in this case is the difference of the simulated values from the interpolated function at the node points).

16.3 Measuring anisotropy with isopotential lines

When current passes through an anisotropic media, the largest potential gradient will be in the direction with the lowest conductivity. This can be readily understood by looking at the expression for the electric potential gradient,

$$\nabla V = -\mathbf{E} = \rho \mathbf{J} = \frac{\mathbf{J}}{\sigma}$$

When the conductivity, σ, decreases, the potential gradient increases. Calculating this expression for non-trivial conductor setups is of course as difficult as ever, so measuring isopotential lines will naturally only be able to point us in the right (current)direction. A few simulations with non-isotropic conductors have been run.

Rod-Electrode Array

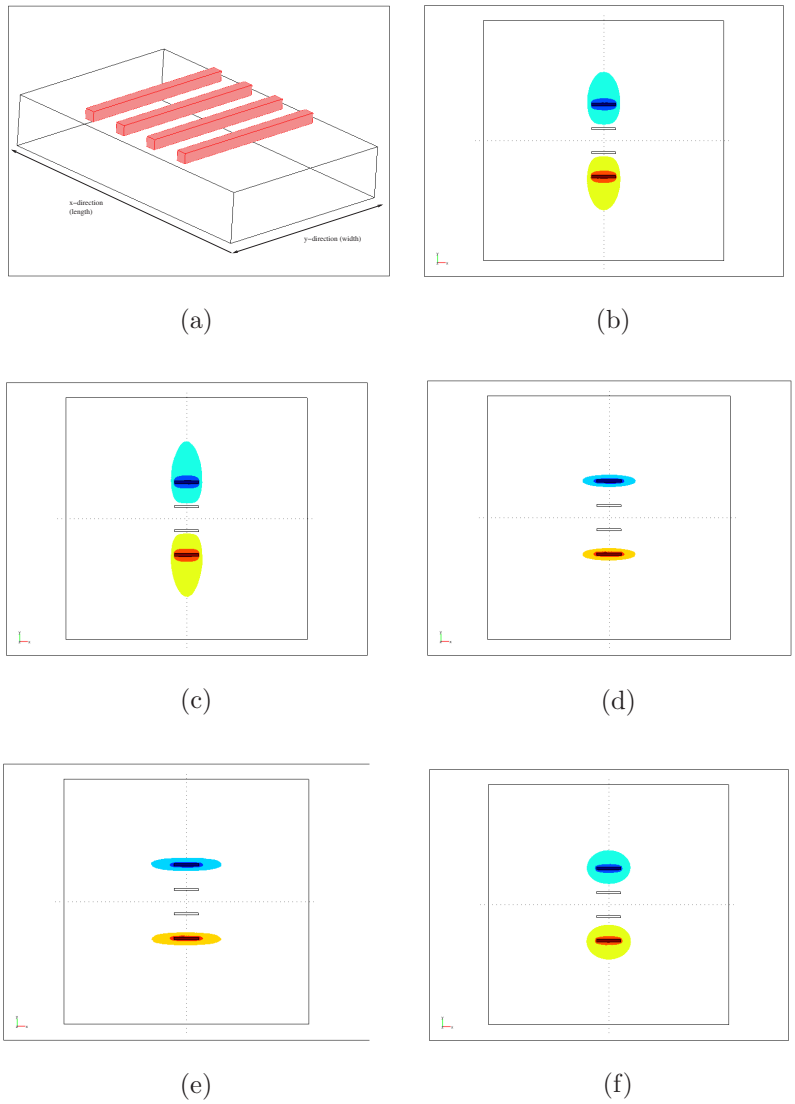
The first batch was done on a cubical conductor with a four-stripe-electrode. The conductor was modelled with 1/8 conductivity in the transversal direction (transverse to the main current direction), 1/16 conductivity in transverse direction, 8/1 conductivity in the parallel direction, 16/1 in the parallel direction and with isotropic conductivity. A plot of the isopotentials is included in figure 16.8.

So what can this tell us about the conductivity tensor? Since the geometry is non-analytic, we will try with a hand-waving argument. In a point-electrode in an infinite medium, the potential would fall as $\frac{1}{r^2}$. It would therefore be probable that the ratio of isopotential lines went as $\frac{a^2}{b^2}$, where a and b are the axes of the potential lines in the high and low conductive direction, respectively.

We see that this is easily found in the transverse high conductive case, but in the parallel case we see that the equipotential lines "flow" out in the parallel direction. One could, however, extrapolate an ellipse around the end corners made of the equipotentials around the electrodes, as demonstrated in 16.9.

Now there are two ellipses on which the semi-major and semi-minor axes can be measured. This has been done by hand, with the results given in the table 2.

We see that there is a good correlation between the conductivity ratio and the square of the axes of the equipotential ellipses. Note, however, that only the conductivity ratio in the xy-plane is taken into consideration in this test-case. Some preliminary simulations were done varying the conductivity



(a) (b) (c) (d) (e) (f)

Figure 16.8: (a) A picture of the model, (b) the isosurfaces of the line-electrode-model with 8/1 conductivity in the parallel direction, (c) 16/1 conductivity in the parallel, (d) 1/8 conductivity in parallel direction, (e) 1/16 conductivity in parallel, (f) isotropic conductivity

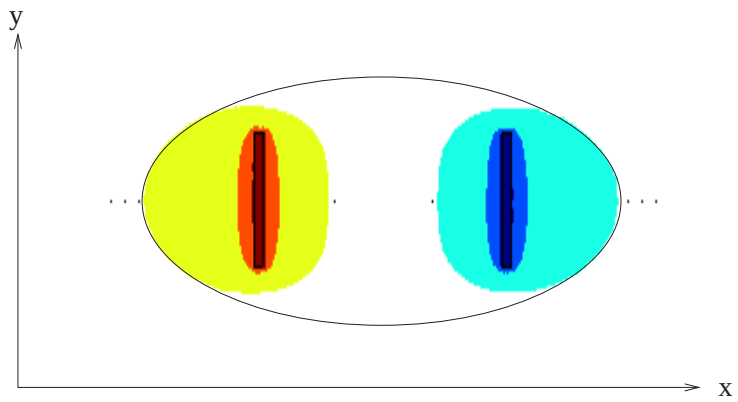


Figure 16.9: An ellipse drawn around the endcaps of the equipotential lines in the case where the high conductivity direction is parallel to the main current direction

Conductivity	$\frac{a^2}{b^2}$
8/1	9
16/1	20
1/8	10.5
1/16	18
1/1	2

Table 2: A table of the anisotropies and the ratios of the related semi-major and semi-minor axes

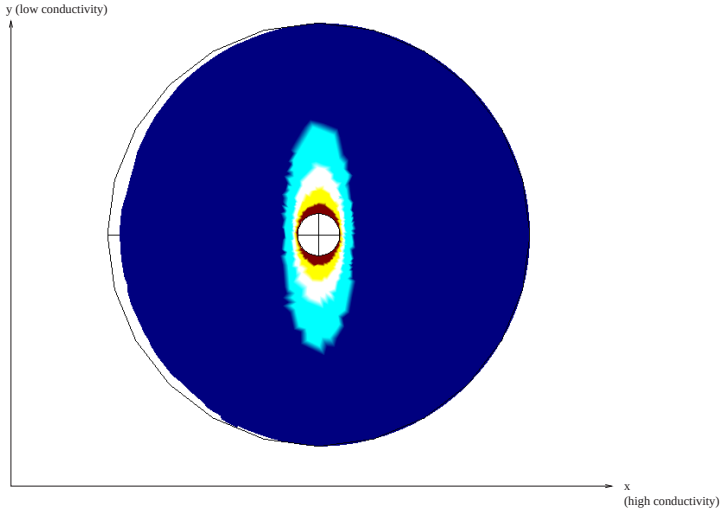


Figure 16.10: A plot of the equipotential lines on a hemispherical conductor with a hemispherical electrode, using the conductor surface-shell as ground

in the z-direction, and this turns out to be just as influential to the potential-distribution in the xy-plane.

Hemispherical case

A simulation on a hemispherical conductor and a center hemispherical electrode was made, with the conductor shell as ground. A plot of the equipotential lines is included in figure 16.10

The ratio of the conductivities is 1/10. The axes measure 11 and 4, which give a squared ratio of 7.5, which is comparable with the results from the stripe-electrodes in the previous paragraph. Again the conductivity in the depth direction (which is the same as that in the y-direction) has not been taken into consideration, and this is can heavily effect potential distribution in the xy-plane.
In other words, simply finding equipotential lines on the surface of a conductor cannot be expected to give unequivocal information on plane conductivity tensor, since we would firstly need to know the component in the z-direction, and secondly know how that effects the distribution in the xy-plane, which will be dependent on many factors, including electrode setup.

17 Electrode Types

In this section the sensitivity and impedance of 4 different electrode types will be analyzed:

- Band electrodes
  The effect of different immersion into the medium will be analyzed.
- Spherical electrodes
  The inter-electrode distance and size of the electrodes will be varied.
- Needle electrodes
  The depth, thickness and inter-electrode distance on bi-polar needle-electrodes will be varied
- Circular band-electrodes
  The sensitivity and impedance of the electrodes will be analyzed with varying inter-electrode distance and band-thickness.

17.1 Band Electrodes

In this case the simulation will be done with a 100x100x100 medium with unity conductivity, and copper band electrodes. These will have a depth and width of 5 with varying immersion into the medium. A model of the electrode setup is included in figure 17.1.

The outer electrodes are current-carrying, the inner are pickup-electrodes.

Boundary Conditions

The boundary conditions are insulating walls of the medium, as well as on the sides of the electrodes and tops of the pickup electrodes (center). On the top of the current-carrying electrodes the current density was set to unity, normal and anti-normal to the top line. The potential was measured on the top line of the pickup-electrodes. On a second model pickup and current carrying electrodes were switched, so that we could check for reciprocity and find a sensitivity plot. A table of the evaluated values is included in table 3.

Calculations

The pickup potentials vary between seemingly random values. This is caused by that the boundary conditions at the current carrying electrodes are set at a constant current, and not a constant potential. Since there is no reference potential (ground) in the medium, the average value of the potential will be

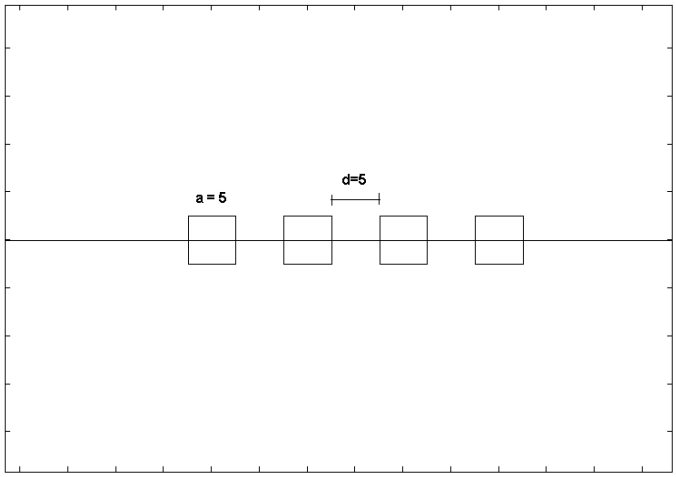


Figure 17.1: A plot of the model for band electrodes. The 5x5 electrodes are copper, the 100x100 medium in which they are immersed has conductivity equal to 1

Table 3: Results of simulation on band-electrode model

Depth	$I(cc)$	$V(pp)_1$	$V(pp)_2$	ΔV	Z
0	-1.00E+01	-76.245576	-118.066194	41.820618	4.1820618
1	-1.00E+01	-215.584216	-257.172123	41.587907	4.1587907
2	-1.00E+01	82.833407	42.506426	40.326981	4.0326981
3	-1.00E+01	-2091.578405	-2130.560422	38.982017	3.8982017
4	-1.00E+01	53.074096	15.460018	37.614078	3.7614078
5	-1.00E+01	30.596507	-5.701972	36.298479	3.6298479
6	-1.00E+01	212.924046	177.804168	35.119878	3.5119878
7	-1.00E+01	214.988261	181.124383	33.863878	3.3863878
8	-1.00E+01	21.72682	-11.077361	32.804181	3.2804181
9	-1.00E+01	19.688819	-12.069374	31.758193	3.1758193
10	-1.00E+01	33.208142	2.827418	30.380724	3.0380724

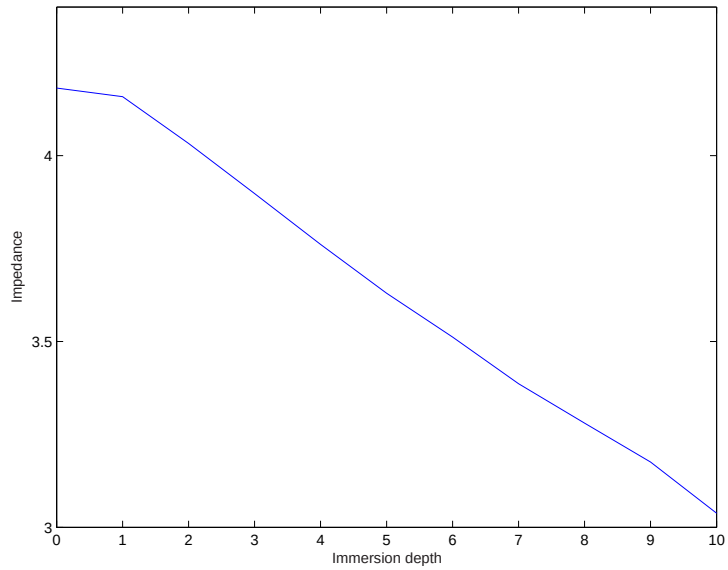


Figure 17.2: A plot of the impedance versus the immersion depth with band electrodes in a semi-infinite medium

floating. The potential difference between the pickup-electrodes, however, is set by the current densities, and has a steadily decreasing value for increasing immersion. A plot of the impedance as a function of immersion depth is included in figure 17.2.

Results

As the figure shows, the impedance drops very quickly. This could be explained by the shortcut-mechanism, that more and more of the current passes through the copper pickup-electrodes. On the one hand, the current-density integral,

$$\Delta V = \int \frac{J}{\sigma}\, dx,$$

decreases as the conductivity between the pickup-electrodes increases. Also, the area of the electrodes increase, quite possibly reducing current density under the electrodes. The top lines of the pickup-electrodes come closer as the electrodes are lowered, and this is where we "listen" (sum the potential). This last effect should be negligible, since the copper electrodes should be considered isopotential. To put it in another way, the integral path, dx, is shortened, but since the conductivity is 7 orders of magnitude larger than

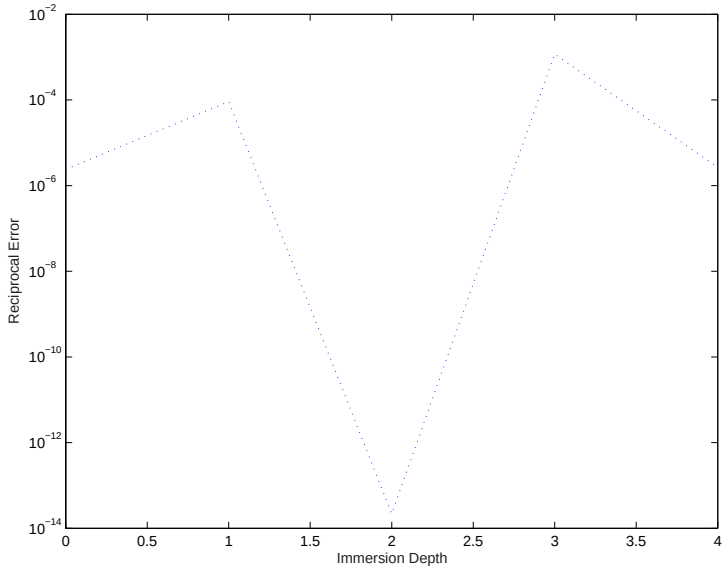


Figure 17.3: A plot of the reciprocal error as a function of immersion depth

the medium, this effect should give no measurable potential difference.

Reciprocity

A calculation of reciprocity was carried out. The impedance is, as in previous sections, first measured as the fraction of pickup-electrodes potential and current-carrying-electrodes current. The setup was switched so that the outer electrodes measured potential and the inner carried the current. The reciprocal impedance, X_R, was measured. The reciprocal error, ϵ, was defined as

$$\epsilon = \frac{Z - Z_R}{Z}$$

A plot of ϵ as a function of immersion is included in figure 17.3.

As can be seen, there seems to be no clear connection between the reciprocity and the immersion depth of the electrodes. The error does, however, seem to be small enough to consider the reciprocity theorem to be valid.

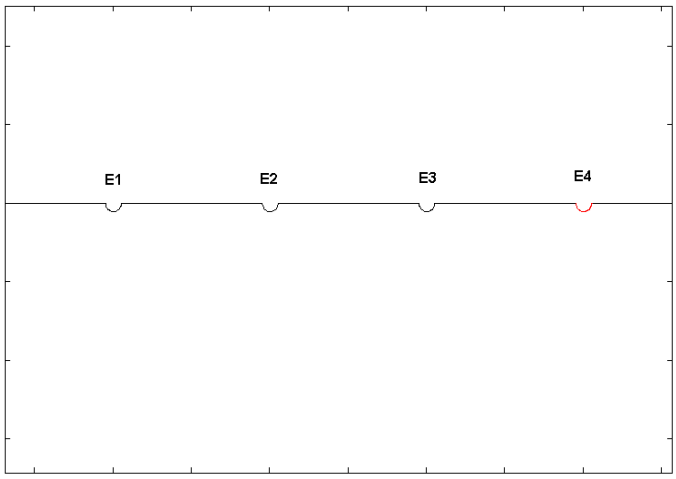


Figure 17.4: A plot of the electrode configuration in the semi-spherical electrodes model

17.2 Semicircular electrodes

Model, Boundary Conditions and Reciprocity

A model with half-sphere indents into a semi-infinite medium was made. The outer indents were set at a constant potential, ϕ_1 and ϕ_2. The potential was then measured at the center indents, considered to be pickup-electrodes. This model has a some problems, since the boundary conditions suggest that the potential appears out of nowhere on the outer indents. A measure of reciprocity was made, giving a reciprocal error of 1.3 %. A plot of the model is included in figure 17.4.

Varying Electrode Radius

When we reduce the electrode size, we have the same amount of current to pass out through a smaller area. This increases current density, and we therefore expect the impedance to rise in the sensitive areas. This is indeed what happens in the simulations, as can be seen in figure 17.5.

There should be no radical difference in the current, since the geometric shape and conductivity map of the conductor is invariant. We could expect a higher impedance owing to a reduction in the shortcut that is the pickup-electrodes. The relationship between the conductivity in the conductor and the electrodes will be significant in how much this shortcut contributes to

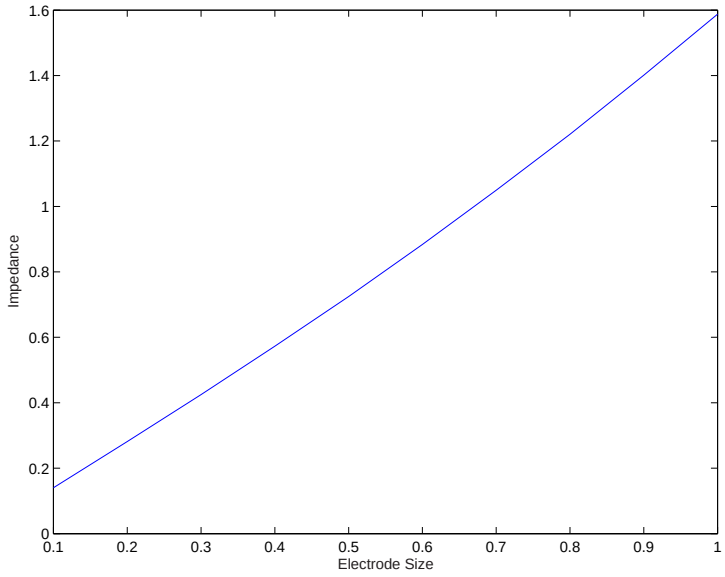


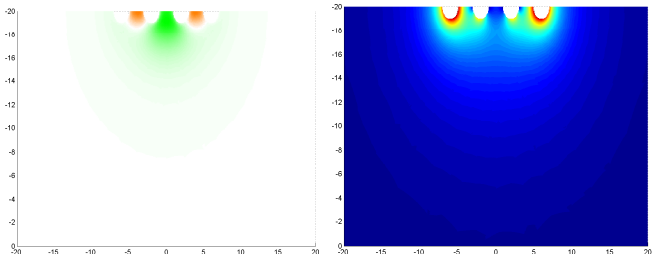
Figure 17.5: The impedance in the usual 2D, copper conductor with circular electrodes, as a function of electrode radius

reduced/increased impedance, as well as the size of the electrodes compared to the geometry of the conductor. However, in this test-case we have used the same conductivity in both the electrodes and the conductor. We therefore expect the shortcut effect to be negligible.

Varying Electrode Distance

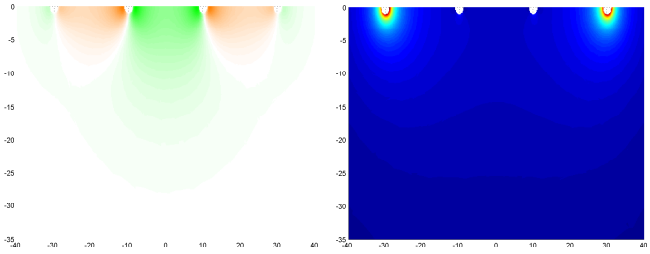
The inter-electrode distance is the other important factor. In the simulation, we varied the interelectrode-distance from 5 to 15, where the total size of the geometry is 100 times 100. The size of the electrodes is kept constant at 1. In the expression of the transfer impedance, there are two factors to be considered. First, there is the ΔV of the pickup electrodes. We do not have much guidance in how this will change. It could be, that the potential-distribution near the boundary remains the same, while there is a sharper potential drop further out. If we look at a sensitivity and an E-field plot of the basic model in figure 17.6, we see that the current density between the outer and the inner electrodes is what we expect to contribute the most to the measured impedance.

If we now look at a similar plot with an electrode spacing five times bigger in figure 17.7 below, we see that we get a lower current density around the area with positive sensitivity.



(a) (b)

Figure 17.6: A plot of the (a) sensitivity distribution and (b) current density of a four-electrode system with inter-electrode spacing of 4 cm



(a) (b)

Figure 17.7: (a) Sensitivity distribution and (b) current density of a four-electrode system with inter-electrode spacing of 20 cm

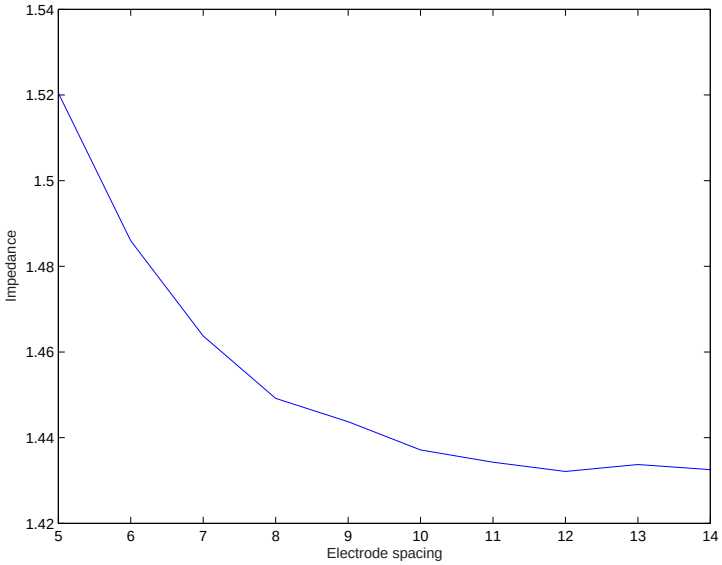


Figure 17.8: The impedance in a four-electrode system vs the electrode spacing

This would imply that we get a reduced impedance as we increase electrode spacing. We have done a loop of simulations increasing the electrode distance for each step, and integrating the ΔV between the pickup electrodes, the current through the current-carrying electrodes and thus the transfer impedance for the system. A plot of the impedance as a function of electrode distance can be found in figure 17.8 below.

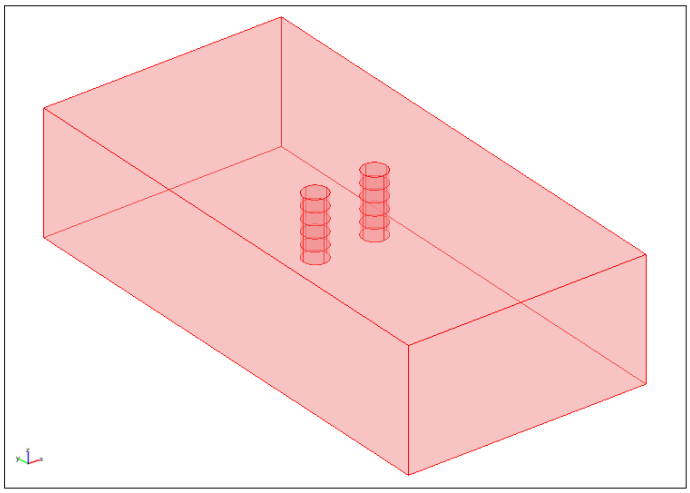


Figure 17.9: A plot of the model used in the needle-electrode model

17.3 Needle Electrodes

Needles are commonly used in many bioimpedance measurements. Here some considerations around the placement of the needles are made.

Model, Boundary Conditions and Reciprocity

Two needles, with length 25, were immersed into a finite medium. The needles had a diameter of 5, and are split into 5 equal segments. Segment number 1,3 and 5, counted from the top, were set as non-conductive ($\sigma = 1^{-14}$). Segments number 2 and 4 were set as copper conductors. In the middle of each conductive segment on the left electrode a point current source was set, with unit current-density. A model with a line current source running through the electrodes was also considered, but added to computation time without significant different results from the point source model.

The medium in which they were immersed was given unit conductivity. The dimensions of the depth, length and width were varied, and the impedance was measured. The potential was measured at the center of the copper segments on the opposite electrode.

The pickup- and current-carrying segments were switched to measure reciprocity, and the error was computed to be around 0.7 %. A plot of the model is included in figure 17.9

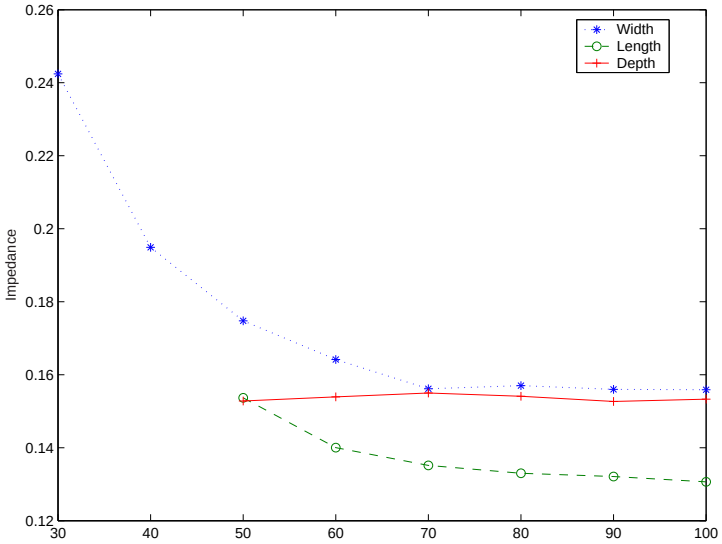


Figure 17.10: A plot of the impedance in a needle-electrode model vs length, width and depth

Impedance vs Length, Width and Depth

The length of the medium (in the x-direction) was varied, as well as the width (y-direction) and depth (z-direction), using a script included in the appendix. All other variables were kept constant, including the inter-electrode distance and the immersion depth. As can be seen in a the figure 17.10, the impedance drops quite dramatically at values when the box walls are close to the electrodes, and then stabilize at a plateau. There is a smaller drop in the width direction, and varying depth causes no discernible effect.

Sensitivity

A plot of the reciprocity in the length direction is included in figure 17.11. The most noticable feature is that there is only negative sensitivity on the outer side of the electrodes. This means that any increase or decrease in resitivity on the outside will cause a reduction or an increase in impedance, respectively. A plot of the sensitivity in the y-direction would reveal that there is very low sensitivity, as could be expected with the low current densities in this direction.

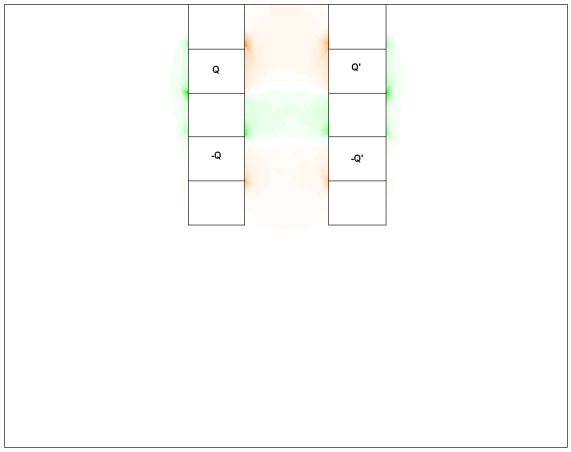


Figure 17.11: A plot of the sensitivity in the length direction on the needle-electrode model

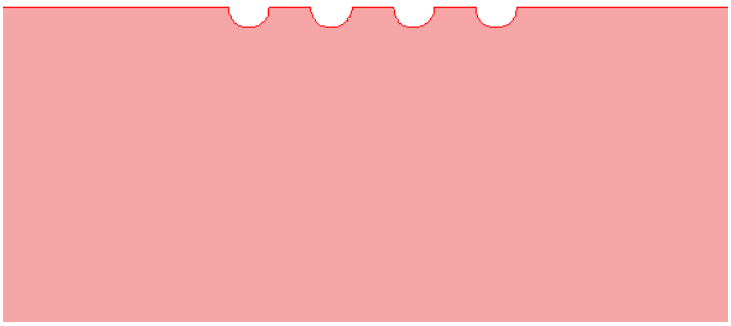


Figure 17.12: A model of the top of a conductor with 4 hemispherical holes in it, used to simulate a surface layer

17.4 A Surface Layer

In some applications there will be a conductive layer in or directly outside the electrodes. A 100×100 model was used, with 4 semicircular holes with radius 1 cut into the top, as can be seen in figure 17.12

The Boundary Conditions

Now we choose a conductive surface layer as the boundary condition on the edges of the holes. The equation of the boundary condition is

$$\mathbf{nJ} = \sigma \frac{(V - V_{ref})}{d}$$

where $\mathbf{n}$ is the normal vector, so that $\mathbf{nJ}$ is the electric flux density at that point. V_{ref} is the reference potential, that we choose. d is the layer thickness.

The conductivity was varied from approximately zero to 20000, and the depth from 0 to 0.8. A script was produced which varied these values, and can be found in the appendix. The results can be seen in figure 17.13

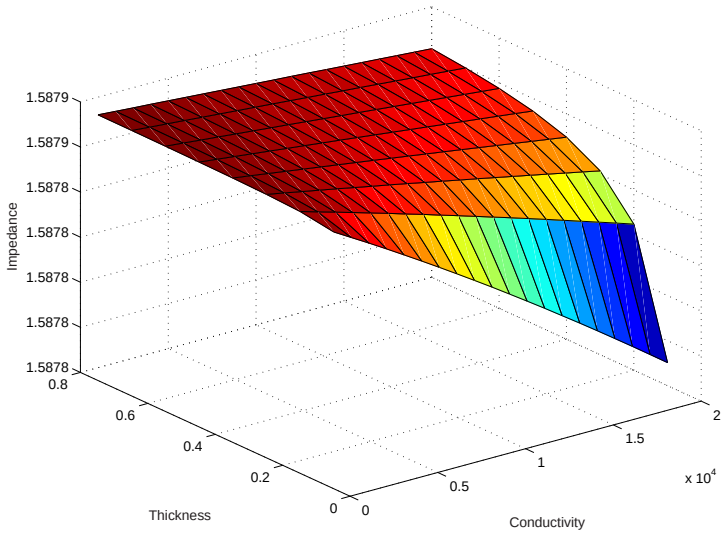


Figure 17.13: A plot of the impedance in a surface layer model vs the conductivity and the thickness of the layer

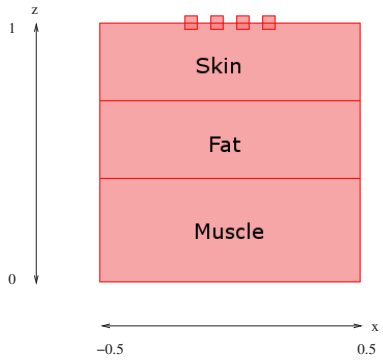


Figure 18.1: A plot of the model used for complex simulations

18 Complex Materials

We finally arrive at the much anticipated complex materials section. As said earlier, virtually everything in this world is more or less dielectric, certainly tissue. For that reason being able to compute complex materials has been of paramount importance.

Restating from the theory-chapter, the conductivity in a conductor with a dielectric component can be stated as

$$\sigma = \sigma' + \mathrm{j}\omega\epsilon'$$

where σ' and ϵ' are the real parts of the conductivity and the permittivity, respectively. ω is the angular frequency of the AC-current. The model used for the complex simulation is the previously seen four-box-electrode array, depicted in figure 18.1.
The model is simulated in two dimensions.

18.1 A Single Layer Model

Boundary Conditions and Materials

First, all the layers in the figure were set to have the same electrical properties, i.e. the permittivity and conductivity of muscle at 50 KHz. These were chosen from [2], and are found in table 18.2. The electrodes are copper.

The boundary conditions contain the crux of the problem. To define the AC-current, a potential or current with a phase factor must be chosen. For

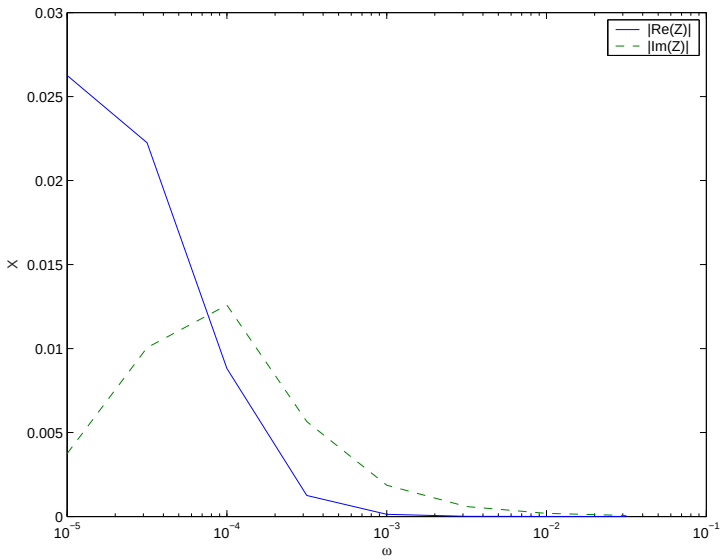


Figure 18.2: A plot of the the real and imaginary parts of the impedance in the complex model

this simulation, a unity current-density of $e^{j\omega}$ on the leftmost electrode was chosen. For the other current-carrying electrode, the condition must be that it is exactly in opposite phase, otherwise there will be a current buildup in the conductor. Therefore the current-density will be opposite, $e^{j\omega+\pi/} = -e^{j\omega}$. The rest of the surface of the model was chosen to be insulating.

Results

The current and the potential were measured at all electrodes, at frequencies varying from low to high. The impedances have been plotted as a function of the frequency in figure 18.2.

As can be seen, both the real and imaginary parts of the impedance are plotted. These can also be compared in a so-called Wessel-plot, in figure 18.3.

18.2 A Multiple Layer Model

Boundary Conditions and Materials

The layers were now given different electrical properties. These were for skin tissue, adipose tissue (fat), and muscle tissue, as depicted in the figure 18.1. To make the computations easy, however, a values were chosen at 50 KHz,

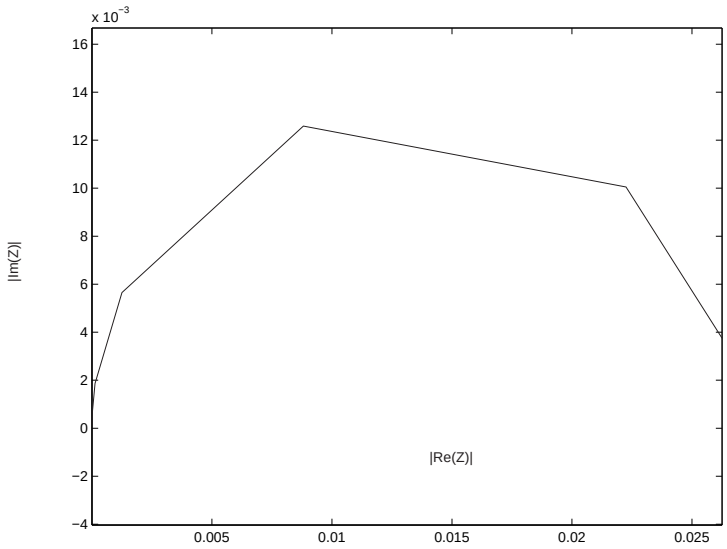


Figure 18.3: A Wessel-plot of the complex material

not varying with the frequency. This is fairly trivial to implement, however, and can be easily included in the script included in the appendix.

The values were the following:

Material	Conductivity	Permittivity
Skin	0.07	1e4
Fat	0.01	30
Muscle	0.7	1e4

The boundary conditions are where the same as the single layer model.

Results

These results can be compared against a case with only one layer, that of a homogenous muscle above. The impedance vs frequency and Wessel-plot can be seen in figures 18.4 and 18.5.

One can see the differences in the potential distribution and electric field streamlines between the two models in figures 18.6 and 18.7, for the multi layer and single layer, respectively.

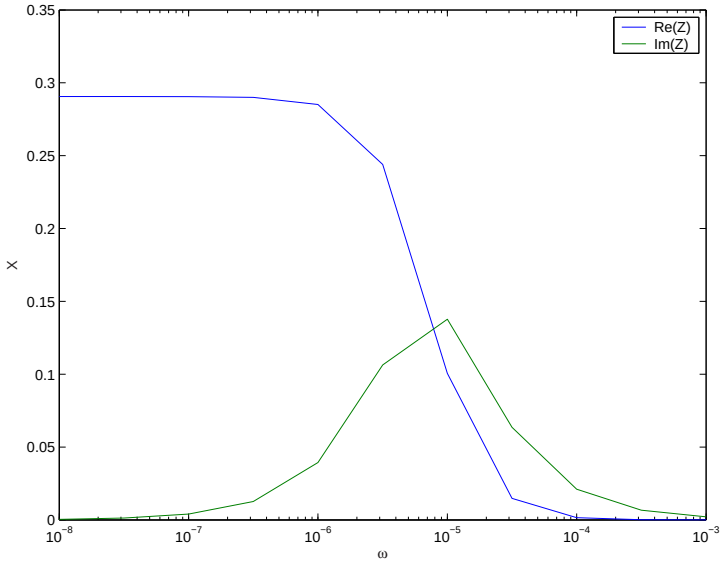


Figure 18.4: A plot of the impedances vs the frequency for a single layer complex model

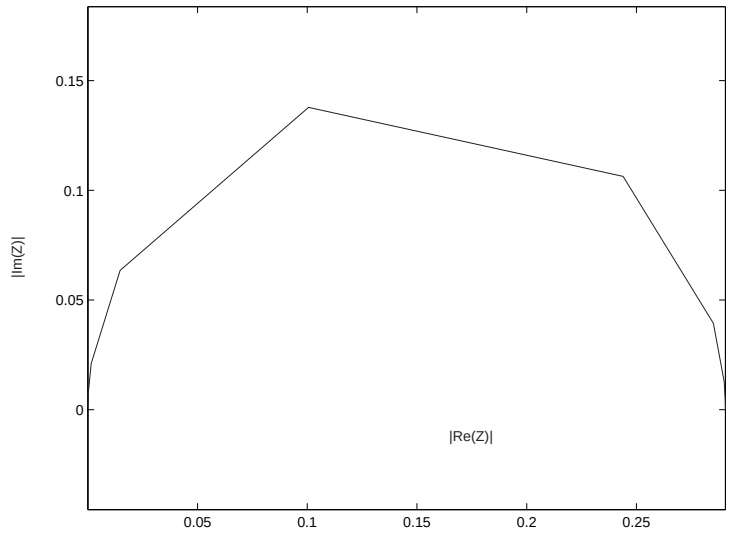


Figure 18.5: A Wessel-plot for a single layer complex model

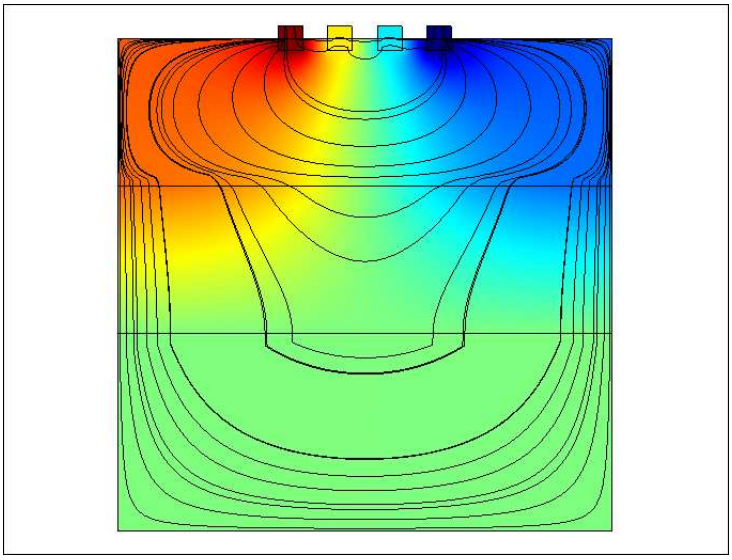


Figure 18.6: A plot of the potential distribution and electric field streamlines in the multi-layer model

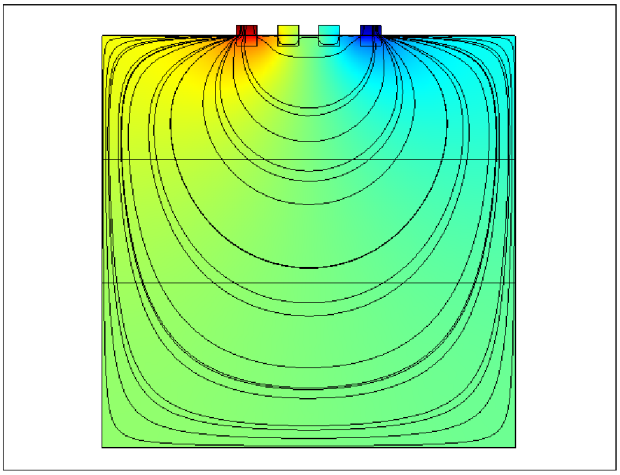


Figure 18.7: A plot of the potential distribution and electric field streamlines in the single-layer model

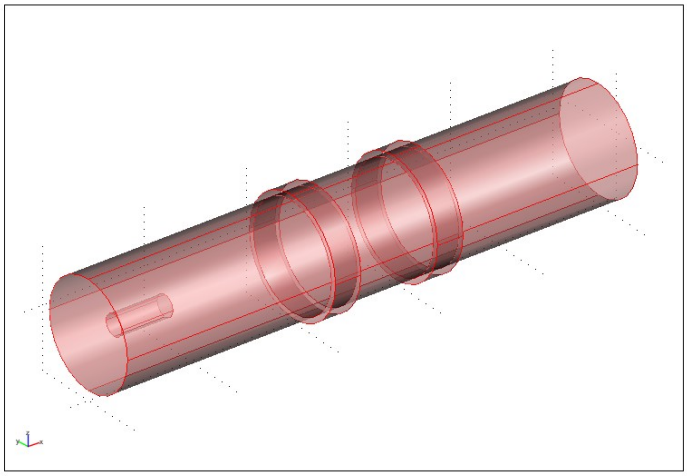


Figure 19.1: A model of the cylinder, with circular electrodes around the middle

19 The Cylinder

An interesting case is the cylinder conductor, which can be considered as a good approximation to for instance an artery or vein. In this simulation the cylinder was 10 long, with a radius of 1. On the outside of this cylinder, two circular electrodes were placed, being 0.5 wide and 0.1 thick. The distance between the electrodes is 2. A plot of the model is included in figure 19.1

In due course, we will place a bolus of high conductivity (copper) to run through the center of the figure, and past the two electrodes, as can be seen in the figure.

19.1 Boundary Conditions

A current is run through the cylinder, with current-density equal to unity at the end surfaces. The electrodes are pickup-electrodes, so they are have continuous interfaces with the cylinder. They do, however, have much larger conductivity than the cylinder (6×10^7 to 1), so we expect a shortcut effect to happen. The rest of the surface is insulating.

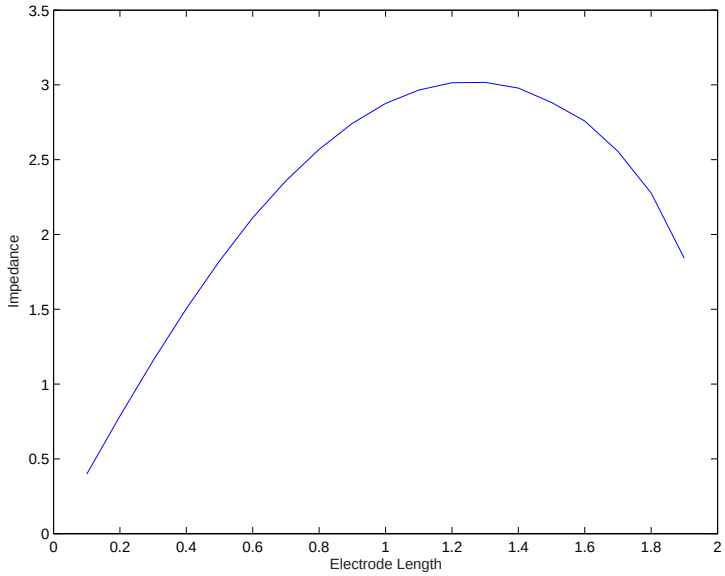


Figure 19.2: The impedance vs the width of the electrodes in the cylinder model

19.2 Results

The model was varied with two variables, electrode width and the progression of the bolus.

Electrode Width

The width of the electrodes was varied from 0.1 to 1.9, where the electrodes would touch at a width of 2.0. The impedance was plotted against the width in figure 19.2.

Bolus Progression

The bolus was passed through the model. As it approaches the pickup-electrodes, it raises the conductivity in an area with high sensitivity, so that the impedance can be expected to fall. A plot of the impedance vs the progression of the bolus is included in figure 19.3

19.3 Axis Symmetry

As can be easily seen, this model is axis-symmetric through the center axis. A 2D-axial symmetric model was also constructed, this can be seen in figure

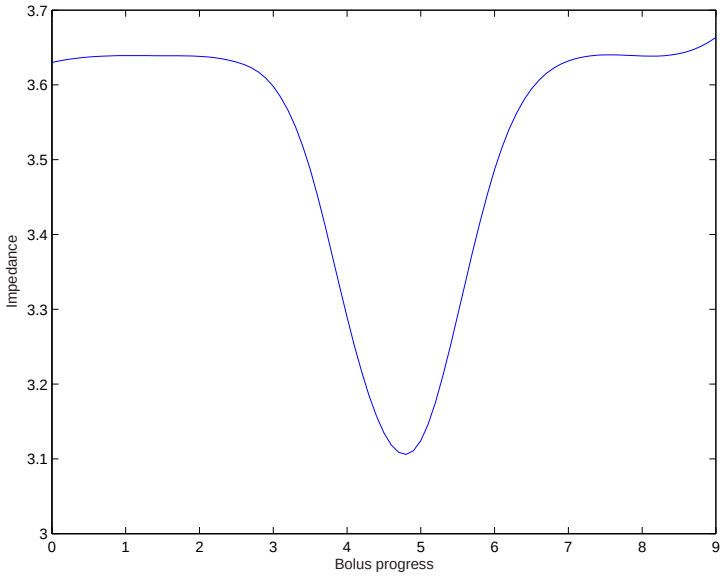


Figure 19.3: The impedance vs progression of a high conductivity bolus in the 3D-cylinder model

19.4.

The same simulation as above regarding the bolus progression was done, wielding the impedance vs progression depicted in figure 19.5

As can be seen, the base impedance is different in the two cases, the relative drop in impedance is also different, 7 % for the 2D-case, and 15 % for the 3D-case. This discrepancy cannot be explained at this point, but I expect the problem might be in the boundaries.

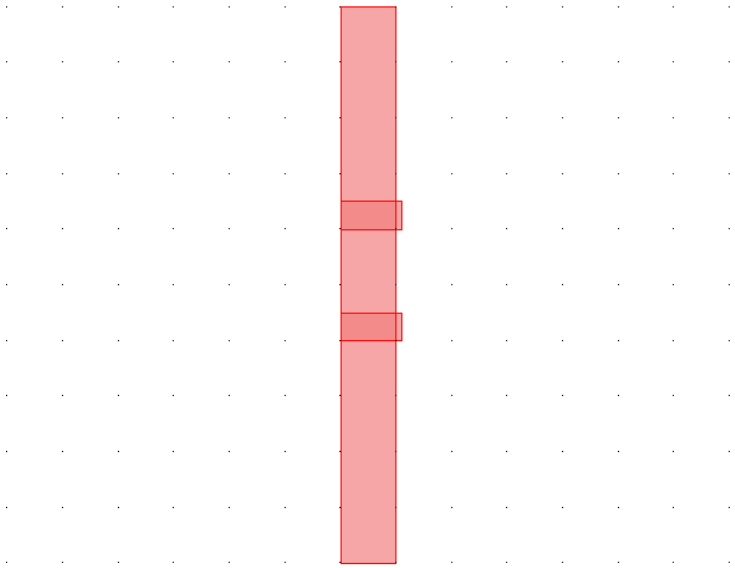


Figure 19.4: The axis-symmetrical, 2D cylinder model

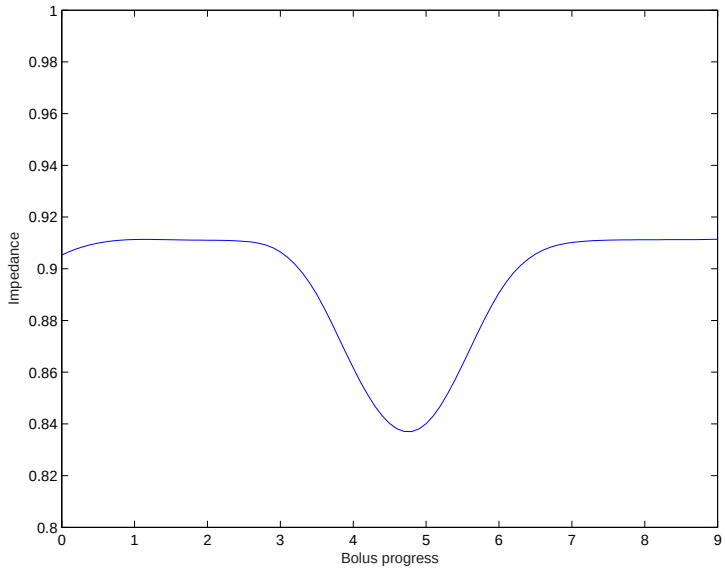


Figure 19.5: Impedance vs progression of bolus in the 2D-cylinder model

Part IV
Conclusion

20 Introduction

In the conclusion I will compare how the results in the thesis compare with the goals I set out with. I will also present the most prominent results, positive and negative. Finally, avenues for further research will be stated.

21 Goals

I will briefly restate the goals from the introduction.

1. Explore the use of the Finite-Element method for computing Electric fields, with special focus on the program Femlab

2. Investigate several problems in the impedance-measurement field

Finite-Element Method

The method for solving partial difference equations taught in computational physics courses (at least at the University of Oslo and Minnesota) is the finite difference method. This is a method which is both easier to grasp mathematically and more widespread in the scientific community. However, the Finite-Element method is gaining popularity, especially because of it ability to handle difficult geometries.

The method was employed with the program Femlab. After a pretty steep learning curve, using Femlab to implement different models was both easy and powerful. The advantages of the program can be summarized in the following:

- Model Design
  The program has an easy to use graphical user interface which basic but sufficient CAD (Computer Assisted Design) tools. Where these are insufficient, models can be imported from more heavy-duty CAD-programs.

- Physics
  All the different equations of the electro-magnetic theory are coded into the program. This has saved a lot of time in implementation.

In stead one can concentrate on searching for correct conditions. The program also has an easy way of connecting different physics, like heat and conductivity.

- Post-processing
  A lot of time has been saved by the built-in tools for post-processing. However, the numbers that have been most interesting, the current in and out of electrodes, as well as the potential at the electrodes, were quite cumbersome to acquire, and should be more easy to access in future releases.

There would be no advantages without disadvantages. To put it briefly, one pays for ease of use with being at the mercy at the inner workings of the programs. The disadvantages can be summarized as;

- Visibility
  Most of the actual computation takes place inside java-code that is not visible to the user. There is therefore quite a bit of uncertainty in what is actually going on. During this investigation, however, most of the errors have naturally been human.

- Scale
  Femlab is limited by the size of the problems it can handle. This is of course a relative size; how big is the biggest detail compared to the smallest. Quite a few times an out-of-memory error was produced. Also, there is no way to make use of computer-clusters for large-scale computations.

In conclusion, Femlab is recommended for ease-of-use, demonstration and solving small-scale or low-detail problems.

Note on Femlab In the end of the thesis Femlab 3.1 was released, which contains code for utilizing 64-bit memory reference, which is advertised to take care of many of the problems encountered on scale problems. We were not able to utilize this function, but it should prove very useful for further study.

The problems

As stated in the last paragraphs, Femlab is very effective for investigating simple problems. The problems solved provide guidance in how different

parameters such as size, shape and anisotropy of the measuring object, and the type, size, placement and immersion of the electrodes.

As usual in the bio-impedance field, no final numbers have been produced, only relative sizes. That is, we are not yet at the point where we can put in all the variables, geometries and materials and receive an impedance that can be expected to correspond with measurements. But the problems demonstrate how different variables will change the impedance, relatively, and in what direction.

Some of the important new results are the following:

- **Simulating Reciprocity**
  The reciprocity theorem has been an invaluable tool in accessing the accuracy of a model; if there was low reciprocity, it would normally mean that something was wrong.

- **The Shortcut-Effect**
  The so-called shortcut effect might very well be the main culprit in many of the cases of lowered (or elevated) impedance. It is of course very hard to say exactly how much the shortcut contributes compared to other effects, but it is now at least clearly visible in the simulations.

- **The Unexpected Impedance Drop**
  In simulating geometry, we have seen that the impedance increases drastically when the boundary of a geometry closes in on the electrodes. This effect has as far as we know never been discussed in articles, and begs explanation.

- **Concentric Electrodes and Anisotropy**
  We can now simulate anisotropy on a microscopic level, which is an important tool in simulating structures like fiber-tissue etc. An important preliminary result is the square-root rule:
  The impedance in a conductor with one conductivity $\sigma_x$in the x-direction, and one conductivity σ_y in the y-direction, has a impedance that varies as $\sqrt{\frac{\sigma_x}{\sigma_y}}$ when using concentric electrodes. This is as far as we know an undocumented effect, and could perhaps be exploited physically.

- **Complex Conductivity / Resistivity**
  The impedance in a complex medium, conductors with complex resistivity or conductivity, has been briefly investigated. The reactance of materials are generally disregarded in literature, thus producing a weak approximation of real bioimpedance.

22 Further Investigation

During this thesis many avenues for further research have been revealed. Some of the most interesting are the following:

- **Complex Materials**
  A brief introduction into this field has begun, but there is much to still be explored. Investigating how sensitivity should be utilized with complex fields at different frequencies is an important topic, as well as combining the complex effects with those of other geometrical-, electrode-, boundary-layer- and anisotropic effects.

- **Time-Varying Sensitivity**
  We have, excluding the cylinder-model, regarded stationary objects in this thesis. An interesting field for further study is to see what happens has materials move and change internal and boundary attributes over time.

- **A Comparative Investigation**
  Many suggestions for effects that affect impedance have been pointed at in this thesis, but not much effort has been made to do a comparison between these. The next step is obviously to try to quantify the relative effect of these.

- **Detail and Size** Last but not least, the level of detail has been a major Achilles' heel of this investigation. To much detail compared to the dimension of the model has lead to undesired effects of the time-consuming and unintelligible-run-time-error kind. Research into how to include higher detail in models, as well as solving the more quickly, is of high importance in every simulation community. Ours should be no exception. Super-computing and clustering will be the only true way to solve real, high-detail physical problems.

23 Bibliography

References

[1] Verifikasjon og anvendelse av en numerisk programpakke brukt til simuleringer i bioimpedans, Vegeir Knudsen, Hovedfagsoppgave ved Fysisk Institutt, Universitetet i Oslo (1999)

[2] Bioimpedance & Bioelectricity Basics, S.Grimnes and Ø.G.Martinsen, Academic Press (2000)

[3] Computational Partial Differential Equations, Hans Petter Langtangen, Springer Verlag (2003)

[4] An Application of Electrocardiographic Lead Theory to Impedance Plethysmography, D.B.Geselowitz, IEEE Transactions on Bio-Medical Engineering, Vol BME-18, No. 1, January 1971

[5] Sears and Zemansky's University Physics 11th Edition with Modern Physics,H.D.Young and R.A.Freedman, Addison Welsey (2004)

24 Appendix

A Matlab code, analytical case

```
%a is the position of the electrode 1, d is the distance between electrode 1 and 2,
%b is the position of the electrode 3, and e is the distance between electrode 3 and
%h is the height

a= 20; d= 20; %constants
b= 60; e= 20;
h = 100;
Dir1 = 2; Dir2=1; % Dirichlet values, or voltages in this case
r1 = 0; %radius from electrode 1
r2 = 0; %radius from electrode 2
r3 = 0; %radius from electrode 3
r4 = 0; %radius from electrode 4

potential = zeros(100,100); % Empty matrix
V1 = zeros(100,100);
V2 = zeros(100,100);

%Loop over all the points
for x = 1:100
    for y = 1:100
        r1 = sqrt((a-x)^2 + (h-y)^2);
        r2 = sqrt((a+d-x)^2 + (h-y)^2);
        r3 = sqrt((b-x)^2 + (h-y)^2 );
        r4 = sqrt((b+e-x)^2 + (h-y)^2);
        V1(y,x) = -Dir2/r1 + Dir2/r4 ;
        V2(y,x) = -Dir1/r2 + Dir1/r3 ;
        potential = V1 + V2 ;
    end
end
figure;
surf(potential);
% Plot the potential
```

B Alternative Matlab code, analytical case

```
%a is the position of the electrode 1, d is the distance between electrode 1 and 2,
%b is the position of the electrode 3, and e is the distance between electrode 3 and
%h is the height
```

```
clear
tic
a= 20; b= 40; %constants
c= 60; d= 80;
h = 100;

k = 10000/pi;

V1 = 2;  % Dirichlet values, or voltages in this case
V2 = 1;
V3 = -1;
V4 = -2;

[X,Y] = meshgrid(1:100);

R1x = a-X; r1 = sqrt(R1x.^2+Y.^2);
R2x = b-X; r2 = sqrt(R2x.^2+Y.^2);
R3x = c-X; r3 = sqrt(R3x.^2+Y.^2);
R4x = d-X; r4 = sqrt(R4x.^2+Y.^2);


Psi1 = V1*r1.^-1 + V4*r4.^-1;
Psi2 = V2*r2.^-1 + V3*r3.^-1;

Potential = Psi1+Psi2;

Jcc_x = (V1*R1x.*r1.^-3 + V4*R4x.*r4.^-3)*k;
Jcc_y = (V1*Y.*r1.^-3 + V4*Y.*r4.^-3)*k;

Jpp_x = (V2*R2x.*r2.^-3 + V3*R3x.*r3.^-3)*k;
Jpp_y = (V2*Y.*r2.^-3 + V3*Y.*r3.^-3)*k;

Sens = Jcc_x.*Jpp_x+ Jcc_y.*Jpp_y;
toc
figure;
surf(Sens,'FaceColor','interp','FaceLighting','phong','EdgeColor','none');
title('sens5');
```

C Code for Diffpack Simulation

Input-file

```
! Input file for the application in 'code'

set gridfile = P=PreproBox | d=2 [0,1]x[0,1] | d=2 e=ElmB4n2D div=[100,100] \
    grading=[1,-2]
set beta = 0.0
set k_const = 1.0
set f_const = 0.0
set Dirichlet value 1 = 2.0
set Dirichlet value 2 = 1.0
set Robin 1 = 1.0
set Robin 2 = 1.0
set axisymmetric = false
set redefine boundary indicators = n=5 \
    names= du/dn=j1 du/dn=j2 du/dn=-j1 du/dn=-j2 du/dn=0 \
    1=() 2=() 3=() 4=() 5=(1 2 3 4)
set add boundary nodes = n=4 \
    b1=[0.199,0.20]x[1,1] b2=[0.399,0.4]x[1,1] \
    b3=[0.599,0.6]x[1,1] b4=[0.799,0.8]x[1,1]
set remove boundary nodes = n=4 \
    b5=[0.199,0.20]x[1,1] b5=[0.399,0.4]x[1,1] \
    b5=[0.599,0.6]x[1,1] b5=[0.799,0.8]x[1,1]
set add material = NONE
.......
```

Main Program

```
#include <Poisson2.h>
#include <readOrMakeGrid.h>
#include <ElmMatVec.h>
#include <FiniteElement.h>


Poisson2:: Poisson2 ()
{
  DPTRACE("Poisson2 constructor");
}

Poisson2:: ~Poisson2 ()
{
  DPTRACE("Poisson2 destructor");
}

void Poisson2:: adm (MenuSystem& menu)  // administer the menu
{
  DPTRACE("Poisson2::adm");
  SimCase::attach (menu); // enables later access to menu
  define (menu);          // define/build the menu
  menu.prompt();          // prompt user, read menu answers into memory
  scan ();                // read menu answers into class variables and init
}

void Poisson2:: define (MenuSystem& menu, int level)
{
  DPTRACE("Poisson2::define");

  menu.addItem (level,          // menu level (1 is main, 2 is first submenu)
                "gridfile",     // menu command/name
                "file or preprocessor command",
                "P=PreproBox | d=2 [0,1]x[0,1] | d=2 e=ElmB4n2D div=[4,4] "
        "grading=[1,1]");

  menu.addItem (level, "beta", "beta*u term in equation", "0.0");
  menu.addItem (level, "k_const", "constant k value used in Poisson2::k",
        "1.0");
  menu.addItem (level, "f_const", "constant f value used in Poisson2::f",
```

```
        "-1.0");
  menu.addItem (level, "Dirichlet value 1", "const u value (ind. 1)", "0.0");
  menu.addItem (level, "Dirichlet value 2", "const u value (ind. 2)", "1.0");
  menu.addItem (level, "Robin u value",
  "alpha coefficient in the u term of the Robin condition (ind. 5)", "0.0");
  menu.addItem (level, "Robin U0 value",
        "constant U0 value in the Robin condition (ind. 5)", "0.0");
  menu.addItem (level, "axisymmetric", "(x_1,x_2) is (r,z)", "false");

  // Instead of supporting all the commands "redefine boundary indicators",
  // "add boundary nodes", etc., in the present class, we could refer the
  // use to the makegrid utility (which supports all the menu items below).
  // However, for multiple loop experiments it is convenient to generate
  // the grids in the simulator rather than manually make a series of grids
  // with, e.g., makegrid first.

  menu.addItem (level, "redefine boundary indicators",
                "GridFE::redefineBoinds(Is) syntax (\"NONE\"=no change)",
                "NONE");
  menu.addItem (level, "add boundary nodes",
                "GridFE::addBoIndNodes(Is) syntax (\"NONE\"=no change)",
        "NONE");
  menu.addItem (level, "remove boundary nodes",
                "GridFE::addBoIndNodes(Is) syntax (\"NONE\"=no change)",
        "NONE");
  menu.addItem (level, "add material",
                "GridFE::addMatrial syntax", "NONE");
  // submenus:
  LinEqAdmFE::defineStatic (menu, level+1);   // linear system parameters
  FEM::defineStatic (menu, level+1);          // numerical integration rule
  SaveSimRes::defineStatic (menu, level+1);   // dumping of fields and curves
}

void Poisson2:: scan ()
{
  DPTRACE("Poisson2::scan");

  MenuSystem& menu = SimCase::getMenuSystem();
  // load answers from the menu:
  String gridfile = menu.get ("gridfile");  // menu.get returns a string
  grid.rebind (new GridFE());  // make an empty grid
```

```
readOrMakeGrid (*grid, gridfile);
dirichlet_val1 = menu.get ("Dirichlet value 1").getReal();
dirichlet_val2 = menu.get ("Dirichlet value 2").getReal();
robin_u        = menu.get ("Robin u value").getReal();
robin_U0       = menu.get ("Robin U0 value").getReal();
beta           = menu.get ("beta").getReal();
k_const        = menu.get ("k_const").getReal();
f_const        = menu.get ("f_const").getReal();
axisymmetric   = menu.get ("axisymmetric").getBool();

String redef   = menu.get ("redefine boundary indicators");
if (!redef.contains("NONE"))  grid->redefineBoInds (redef);
String addbn   = menu.get ("add boundary nodes");
if (!addbn.contains("NONE"))  grid->addBoIndNodes (addbn, ON);
addbn          = menu.get ("remove boundary nodes");
if (!addbn.contains("NONE"))  grid->addBoIndNodes (addbn, OFF);
String addmat  = menu.get ("add material");
if (!addmat.contains("NONE")) grid->addMaterial (addmat);

FEM::scan (menu);
database.rebind (new SaveSimRes ());
database->scan (menu, grid->getNoSpaceDim());

// when running multiple loops, a statement like u.rebind(new FieldFE...)
// *first* creates a new FieldFE object and then detaches the u pointer
// from the old object such that this object can be delete, but at a
// point of time the old and new object exist side by side.
// to avoid such memory use, one can apply the construction
// u.detach().rebind (new FieldFE...)
// now, the detach function will detach the pointer such that the
// old field can be deleted before the new one is created

// allocate data structures in the class:
u.detach().rebind (new FieldFE  (*grid, "u"));
flux.detach().rebind (new FieldsFE (*grid, "flux"));  // vector field
flux_magnitude.detach().rebind (new FieldFE (*grid, "flux_magnitude"));
dof.detach().rebind (new DegFreeFE (*grid, 1));   // 1 for 1 unknown per no
lineq.detach().rebind (new LinEqAdmFE()); // linear system and solvers
lineq->scan (menu);                       // determine storage and solver t
linsol.redim (dof->getTotalNoDof());      // init length of lin.sys. soluti
lineq->attach (linsol);                   // use linsol as sol.vec. in line
```

```
  u_anal.detach().rebind (new FieldFE (*grid, "u_anal"));
}

void Poisson2:: fillEssBC ()
{
  DPTRACE("Poisson2::fillEssBC");
  // boundary indicator convention:
  // bo.ind. 1: constant Dirichlet condition 1
  // bo.ind. 2: constant Dirichlet condition 2
  // bo.ind. 3: variable Dirichlet condition (g function)
  // bo.ind. 4: homogeneous Neumann condition
  // bo.ind. 5: Robin condition

  dof->initEssBC ();                    // init for assignment below
  const int nno = grid->getNoNodes() ;  // no of nodes
  Ptv(real) x;                          // a nodal point
  for (int i = 1; i <= nno; i++) {
    // is node i subjected to any Dirichlet value boundary indicator?
    if (grid->boNode (i, 1))
      dof->fillEssBC (i, -dirichlet_val1);
    if (grid->boNode (i, 2))
      dof->fillEssBC (i, -dirichlet_val2);
    if (grid->boNode (i, 3))
      dof->fillEssBC (i, dirichlet_val2);
    if (grid->boNode (i, 4))
      dof->fillEssBC (i, dirichlet_val1);
  }
#ifdef DP_DEBUG
   dof->printEssBC (s_o, 2);  // for checking the essential boundary cond.
#endif
}

void Poisson2:: calcElmMatVec
  (int elm_no, ElmMatVec& elmat, FiniteElement& fe)
{
  DPTRACE("Poisson2:: calcElmMatVec");

  // integral over element:
  FEM::calcElmMatVec (elm_no, elmat, fe);

  // integral over the boundaries (here: Robin condition)
```

```
  int s, nsides = fe.getNoSides();
  for (s = 1; s <= nsides; s++) {
    if (fe.boSide (s, 5))   // condition on this side (boundary ind. 5)?
      numItgOverSide (s, 5, elmat, fe);
  }
#if DP_DEBUG >= 2
  elmat.print (s_o);
#endif
}

void Poisson2:: integrands (ElmMatVec& elmat, const FiniteElement& fe)
{
  real f_value = f (fe);
  real k_value = k (fe);
  int i,j,q;
  const int nbf = fe.getNoBasisFunc(); // no of nodes (or basis functions)
  real detJxW = fe.detJxW();     // det J times numerical itg.-weight
  if (axisymmetric) detJxW *= fe.getGlobalEvalPt()(1);
  const int nsd = fe.getNoSpaceDim();

  real gradNi_gradNj;
  for (i = 1; i <= nbf; i++) {
    for (j = 1; j <= nbf; j++) {
      gradNi_gradNj = 0;
      for (q = 1; q <= nsd; q++)
        gradNi_gradNj += fe.dN(i,q) * fe.dN(j,q);

      elmat.A(i,j) += (beta*fe.N(i)*fe.N(j) + k_value*gradNi_gradNj)*detJxW;
    }
    elmat.b(i) += fe.N(i)*f_value*detJxW;
  }
}

void Poisson2:: integrands4side
  (int /*side*/, int boind, ElmMatVec& elmat, const FiniteElement& fe)
{
  const int nbf = fe.getNoBasisFunc();
  real detSideJxW = fe.detSideJxW();
  if (axisymmetric) detSideJxW *= fe.getGlobalEvalPt()(1);
  int i,j;
```

```
  if (boind == 5) {
    for (i = 1; i <= nbf; i++) {
      elmat.b(i) += fe.N(i)*robin_U0*detSideJxW;
      for (j = 1; j <= nbf; j++)
    elmat.A(i,j) += robin_u*fe.N(i)*fe.N(j)*detSideJxW;
    }
  }
}


void Poisson2:: solveProblem () // main routine of class Poisson2
{
  DPTRACE("Poisson2::solveProblem");

  s_o << "\nNo of nodes: " << grid->getNoNodes() << ", no of elements: "
      << grid->getNoElms() << ", element: " << grid->getElmType(1) << "\n";

  fillEssBC ();                // set essential boundary conditions
  makeSystem (*dof, *lineq);   // calculate linear system

#if DP_DEBUG >= 2
  if (grid->getNoElms() <= 50) // print linear system after assembly
    lineq->debugPrint(s_o, true);
#endif

  linsol.fill (0.0);           // set all entries to 0 in start vector
  dof->insertEssBC (linsol);   // insert boundary values in start vector
  lineq->solve();              // solve linear system
  int niterations; bool c;     // for iterative solver statistics
  if (lineq->getStatistics(niterations,c))  // iterative solver?
    s_o << oform("  solver%sconverged in %3d iterations\n",
                 c ? " " : " not ",niterations);

  // the solution is now in linsol, it must be copied to the u field:
  dof->vec2field (linsol, *u);

  // compute flux = -k*grad(u) :
  FEM::makeFlux   (*flux, *u);         // (calls the k function)

  saveResults();
```

```
#if DP_DEBUG >= 2
  u->values().print(s_o,"u->values()");
#endif
}

void Poisson2:: saveResults ()
{
  database->dump (*u);          // dump u for later visualization
  database->lineCurves (*u);    // dump u along lines for later plotting
  database->dump  (*flux);
  flux->magnitude (*flux_magnitude);   // compute norm of flux at each node
  database->dump  (*flux_magnitude);
}

void Poisson2:: resultReport () {}

real Poisson2:: f (const FiniteElement& /*fe*/, real /*t*/)
     { return f_const; }
real Poisson2:: g (const Ptv(real)& /*x*/) { return  0; }

// The k function is used in integrands in defining the weak form *and* in
// FEM::makeFlux when computing the flux -k*grad(u). The FEM::makeFlux
// function requires that k here in the solver class has exactly the
// following signature (real return value and FiniteElement and real argument
real Poisson2:: k (const FiniteElement& /*fe*/, real /*t*/)
     { return  k_const; }
// k, f, and g are virtual and can be overrided in subclasses of Poisson2
```

Header file

```
#ifndef Poisson2_h_IS_INCLUDED  // prevent multiple inclusions of
#define Poisson2_h_IS_INCLUDED  // this header file

#include <FEM.h>           // FEM algorithms, FieldFE, GridFE etc
#include <DegFreeFE.h>     // mapping: nodal values -> unknowns in linear sys
#include <LinEqAdmFE.h>    // linear systems, storage and solution
#include <SaveSimRes.h>    // storage tool for later visualization
//#include <gdbprint.h>    // for printing Diffpack objects in gdb (debugger)

class Poisson2 : public FEM
{
```

```
public:
  // general data:
  Handle(GridFE)     grid;   // finite element grid
  Handle(DegFreeFE)  dof;    // trivial mapping here: nodal values -> unknowns
  Handle(FieldFE)    u;      // finite element field, the primary unknown
  Vec(real)          linsol; // solution of linear system
  Handle(LinEqAdmFE) lineq;  // linear system, storage and solution
  Handle(SaveSimRes) database;         // store computed fields on file
  Handle(FieldsFE)   flux;             // flux = -k*grad(u)
  Handle(FieldFE)    flux_magnitude;   // = ||flux||
  real               beta;             // PDE is -div(k*grad(u) + beta*u = f
  real               k_const;          // const k value used in k func.
  real               f_const;          // const f value used in f func.
  real dirichlet_val1;       // constant u values at the boundary
  real dirichlet_val2;       // constant u values at the boundary
  real robin_u, robin_U0;    // constants in Robin boundary cond.
  bool axisymmetric;         // true: (r,z) cylindrical coordinates

  virtual void calcElmMatVec  // needed here because of the boundary integral
    (int elm_no, ElmMatVec& elmat, FiniteElement& fe);

  virtual void integrands4side  // integrand in boundary integral
    (int side, int boind, ElmMatVec& elmat, const FiniteElement& fe);

  virtual void fillEssBC ();           // set boundary conditions

  virtual void integrands  // evaluate weak form in the FEM equations
    (ElmMatVec& elmat, const FiniteElement& fe);


  Poisson2 ();
  virtual ~Poisson2 ();

  virtual void adm    (MenuSystem& menu);
  virtual void define (MenuSystem& menu, int level = MAIN);
  virtual void scan   ();
  virtual void solveProblem ();        // main driver routine
  virtual void resultReport ();
  virtual void saveResults  ();        // dump solution to file

  // source term in the PDE:
```

```
  virtual real f (const FiniteElement& fe, real t = DUMMY);
  // coefficient in the diffusion term (also required by FEM::makeFlux):
  virtual real k (const FiniteElement& fe, real t = DUMMY);
  // essential boundary conditions:
  virtual real g (const Ptv(real)& x);

};
#endif
```

D Femlab-script for DC-case

```
% FEMLAB Model M-file
% Generated 06-Jan-2004 18:26:40 by FEMLAB 2.3.0.145.

flclear fem
% FEMLAB Version
clear vrsn;
vrsn.name='FEMLAB 2.3';
vrsn.major=0;
vrsn.build=145;
fem.version=vrsn;

% Recorded command sequence

% New geometry 1
fem.sdim={'x','y'};

% Geometry
clear s c p
R1=rect2(0,1,0,1,0);
objs={R1};
names={'R1'};
s.objs=objs;
s.name=names;

x=[0.19900000000000001 0.20000000000000001];
y=[1 1];
B1=curve2(x,y);
x=[0.39900000000000002 0.40000000000000002];
y=[1 1];
B2=curve2(x,y);
x=[0.59899999999999998 0.59999999999999998];
y=[1 1];
B3=curve2(x,y);
x=[0.79900000000000004 0.80000000000000004];
y=[1 1];
B4=curve2(x,y);
objs={B1,B2,B3,B4};
names={'B1','B2','B3','B4'};
c.objs=objs;
```

```
c.name=names;

objs={};
names={};
p.objs=objs;
p.name=names;

drawstruct=struct('s',s,'c',c,'p',p);
fem.draw=drawstruct;
fem.geom=geomcsg(fem);

clear appl

% Application mode 1
appl{1}.mode=flpdedc2d('dim',{'V'},'sdim',{'x','y'},'submode','std','tdiff',
'on');
appl{1}.dim={'V'};
appl{1}.form='coefficient';
appl{1}.border='off';
appl{1}.name='dc';
appl{1}.var={};
appl{1}.assign={'E';'E';'Ex';'Ex';'Ey';'Ey';'J';'J';'Jx';'Jx';'Jy';'Jy';'Q';
'Q';'nJ';'nJ'};
appl{1}.elemdefault='Lag2';
appl{1}.shape={'shlag(2,''V'')'};
appl{1}.sshape=2;
appl{1}.equ.sigma={{{'1.0'}}};
appl{1}.equ.Q={'0'};
appl{1}.equ.gporder={{4}};
appl{1}.equ.cporder={{2}};
appl{1}.equ.shape={1};
appl{1}.equ.init={{{'0'}}};
appl{1}.equ.usage={1};
appl{1}.equ.ind=1;
appl{1}.bnd.g={'0','0','0','0','0'};
appl{1}.bnd.q={'0','0','0','0','0'};
appl{1}.bnd.V={'0','-v1','-v2','v2','v1'};
appl{1}.bnd.type={'qg0','V','V','V','V'};
appl{1}.bnd.gporder={{0},{0},{0},{0},{0}};
appl{1}.bnd.cporder={{0},{0},{0},{0},{0}};
appl{1}.bnd.shape={0,0,0,0,0};
```

```
appl{1}.bnd.ind=[1 1 1 2 1 3 1 4 1 5 1 1];

fem.appl=appl;

% Initialize mesh
fem.mesh=meshinit(fem,...
'Out',    {'mesh'},...
'jiggle', 'mean',...
'Hcurve', 0.29999999999999999,...
'Hgrad',  1.3,...
'Hmax',   {[],zeros(1,0),zeros(1,0),zeros(1,0)},...
'Hnum',   {[],zeros(1,0)},...
'Hpnt',   {10,zeros(1,0)});

% Differentiation rules
fem.rules={};

% Problem form
fem.outform='coefficient';

% Differentiation
fem.diff={'expr'};

% Differentiation simplification
fem.simplify='on';

% Boundary conditions
clear bnd
bnd.g={'0','0','0','0','0'};
bnd.q={'0','0','0','0','0'};
bnd.V={'0','-v1','-v2','v2','v1'};
bnd.type={'qg0','V','V','V','V'};
bnd.gporder={{0},{0},{0},{0},{0}};
bnd.cporder={{0},{0},{0},{0},{0}};
bnd.shape={0,0,0,0,0};
bnd.ind=[1 1 1 2 1 3 1 4 1 5 1 1];
fem.appl{1}.bnd=bnd;

% PDE coefficients
clear equ
equ.sigma={{{'1.0'}}};
```

```
equ.Q={'0'};
equ.gporder={{4}};
equ.cporder={{2}};
equ.shape={1};
equ.init={{{'0'}}};
equ.usage={1};
equ.ind=1;
fem.appl{1}.equ=equ;

% Internal borders
fem.appl{1}.border='off';

% Shape functions
fem.appl{1}.shape={'shlag(2,''V'')'};

% Geometry element order
fem.appl{1}.sshape=2;

% Define constants
fem.const={...
'v1',     2,...
'v2',     1};

% Multiphysics
fem=multiphysics(fem);

% Extend the mesh
fem.xmesh=meshextend(fem,'context','local','cplbndeq','on','cplbndsh','on');

% Evaluate initial condition
init=asseminit(fem,...
'context','local',...
'init',   fem.xmesh.eleminit);

% Solve problem
fem.sol=femlin(fem,...
'jacobian','equ',...
'out',    {'sol'},...
'init',   init,...
'context','local',...
'sd',     'off',...
```

```
'nullfun','flnullorth',...
'blocksize',5000,...
'solcomp',{'V'},...
'linsolver','matlab',...
'method', 'eliminate',...
'uscale', 'auto');

% Save current fem structure for restart purposes
fem0=fem;

% Plot solution
postplot(fem,...
'geomnum',1,...
'context','local',...
'tridata',{'V','cont','internal'},...
'trifacestyle','interp',...
'triedgestyle','none',...
'trimap', 'jet',...
'trimaxmin','off',...
'tribar', 'on',...
'geom',   'on',...
'geomcol','bginv',...
'refine', 3,...
'contorder',2,...
'phase',  0,...
'title',  'Surface: electric potential (V)  ',...
'renderer','zbuffer',...
'solnum', 1,...
'axisvisible','on')

% Plot solution
postplot(fem,...
'geomnum',1,...
'context','local',...
'tridata',{'V','cont','internal'},...
'trifacestyle','interp',...
'triedgestyle','none',...
'trimap', 'jet',...
'trimaxmin','off',...
'tribar', 'on',...
'geom',   'on',...
```

```
'geomcol','bginv',...
'refine', 3,...
'contorder',2,...
'phase',  0,...
'axisvisible','on',...
'title',  'Surface: electric potential (V)  ',...
'renderer','zbuffer',...
'solnum', 1)

% Differentiation rules
fem.rules={};

% Problem form
fem.outform='coefficient';

% Differentiation
fem.diff={'expr'};

% Differentiation simplification
fem.simplify='on';

% Boundary conditions
clear bnd
bnd.g={'0','0','0','0','0'};
bnd.q={'0','0','0','0','0'};
bnd.V={'0','-v1','-v2','v2','v1'};
bnd.type={'qg0','V','V','V','V'};
bnd.gporder={{0},{0},{0},{0},{0}};
bnd.cporder={{0},{0},{0},{0},{0}};
bnd.shape={0,0,0,0,0};
bnd.ind=[1 1 1 2 1 3 1 4 1 5 1 1];
fem.appl{1}.bnd=bnd;

% PDE coefficients
clear equ
equ.sigma={{{'1.0'}}};
equ.Q={'0'};
equ.gporder={{4}};
equ.cporder={{2}};
equ.shape={1};
equ.init={{{'0'}}};
```

```
equ.usage={1};
equ.ind=1;
fem.appl{1}.equ=equ;

% Internal borders
fem.appl{1}.border='off';

% Shape functions
fem.appl{1}.shape={'shlag(2,''V'')'};

% Geometry element order
fem.appl{1}.sshape=2;

% Define constants
fem.const={...
'v1',     2,...
'v2',     1};

% Multiphysics
fem=multiphysics(fem);

% Extend the mesh
fem.xmesh=meshextend(fem,'context','local','cplbndeq','on','cplbndsh','on');

% Evaluate initial condition
init=asseminit(fem,...
'context','local',...
'init',   fem.xmesh.eleminit);

% Solve problem
fem.sol=femlin(fem,...
'jacobian','equ',...
'out',    {'sol'},...
'init',   init,...
'context','local',...
'sd',     'off',...
'nullfun','flnullorth',...
'blocksize',5000,...
'solcomp',{'V'},...
'linsolver','matlab',...
'method', 'eliminate',...
```

```
'uscale', 'auto');

% Save current fem structure for restart purposes
fem0=fem;

% Plot solution
postplot(fem,...
'geomnum',1,...
'context','local',...
'tridata',{'V','cont','internal'},...
'trifacestyle','interp',...
'triedgestyle','none',...
'trimap', 'jet',...
'trimaxmin','off',...
'tribar', 'on',...
'geom',   'on',...
'geomcol','bginv',...
'refine', 3,...
'contorder',2,...
'phase',  0,...
'axisvisible','on',...
'title',  'Surface: electric potential (V)  ',...
'renderer','zbuffer',...
'solnum', 1)
```

E Script for calculating impedance from conductors of varying length

```
function imp = impedance(length)
% FEMLAB Model M-file
% Generated by FEMLAB 3.0a (FEMLAB 3.0.0.228, $Date: 2004/04/05 18:04:31 $)

flclear fem

% Femlab version
clear vrsn
vrsn.name = 'FEMLAB 3.0';
vrsn.ext = 'a';
vrsn.major = 0;
vrsn.build = 228;
vrsn.rcs = '$Name:  $';
vrsn.date = '$Date: 2004/04/05 18:04:31 $';
fem.version = vrsn;

% Geometry
g1=circ2('.08','base','center','pos',{'.2','1'},'rot','0');
garr=geomarrayr(g1,.2,0,4,1);
[g2,g3,g4,g5]=deal(garr{:});
g6=rect2(length,'1','base','center','pos',{'0.5','0.5'},'rot','0');
clear s
s.objs={g1,g3,g4,g5,g6};
s.name={'C1','C2','C3','C4','R1'};
s.tags={'g1','g3','g4','g5','g6'};

fem.draw=struct('s',s);
fem.geom=geomcsg(fem);

% Initialize mesh
fem.mesh=meshinit(fem,'report','off');

% (Default values are not included)

% Application mode 1
clear appl
appl.mode.class = 'ConductiveMediaDC';
```

```
appl.border = 'on';
appl.assignsuffix = '_dc';
clear bnd
bnd.V0 = {0,0,2,0};
bnd.type = {'cont','nJ0','V','V'};
bnd.ind = [2,2,2,1,2,1,2,1,2,1,2,2,3,2,3,2,1,2,1,2,1,2,1,2,4,2,4,2];
appl.bnd = bnd;
fem.appl{1} = appl;

% Multiphysics
fem=multiphysics(fem);

% Extend mesh
fem.xmesh=meshextend(fem);

% Solve problem
fem.sol=femlin(fem, ...
               'solcomp',{'V'}, ...
               'outcomp',{'V'}, ...
               'nonlin','off', ...
               'report','off');

% Save current fem structure for restart purposes
fem0=fem;

%Plot solution
postplot(fem, ...
          'tridata',{'V','cont','internal'}, ...
          'trimap','jet(1024)', ...
          'title','Surface: Electric potential', ...
          'refine',3);%, ...
          'axis',[-1.5536516915557996,1.5536516915557996,-0.9058544910627784,

% Integrate
I1=postint(fem,'nJ_dc', ...
           'solnum',[[1]], ...
           'phase',0*pi/180, ...
           'edim',1, ...
           'intorder',4, ...
           'geomnum',1, ...
           'dl',[25,27]);
```

```
% Integrate
I2=postint(fem,'nJ_dc', ...
           'solnum',[[1]], ...
           'phase',0*pi/180, ...
           'edim',1, ...
           'intorder',4, ...
           'geomnum',1, ...
           'dl',[13,15]);

% Integrate
V1=postint(fem,'V', ...
           'solnum',[[1]], ...
           'phase',0*pi/180, ...
           'edim',1, ...
           'intorder',4, ...
           'geomnum',1, ...
           'dl',[17,18]);

% Integrate
V2=postint(fem,'V', ...
           'solnum',[[1]], ...
           'phase',0*pi/180, ...
           'edim',1, ...
           'intorder',4, ...
           'geomnum',1, ...
           'dl',[21,23]);

% Integrate Area
Area=postint(fem,'1', ...
           'solnum',[[1]], ...
           'phase',0*pi/180, ...
           'edim',1, ...
           'intorder',4, ...
           'geomnum',1, ...
           'dl',[21,23]);


Imedian = (I1-I2)/2;
DeltaV = (V1-V2);
```

```
DeltaV = DeltaV/Area;
imp = Imedian/DeltaV;
```

F Script for Simulating a Surface Layer

```
% In main
tic nThickSteps = 1; nRestSteps = 1; nSizes = 10; hBar = waitbar(0,
'Computing');


for i = 1:nThickSteps
    for j = 1: nRestSteps
        for k = 1:nSizes

    conductivity = i*1000;
    thickness = j*0.1;
    size = 4 + k;

    fem = conducting_layer_1mm_real_solved(thickness, conductivity, size);

    % Integrate
    I1(k)=postint(fem,'nJ_dc', ...
        'solnum',1, ...
        'phase',0*pi/180, ...
        'edim',1, ...
        'intorder',4, ...
        'geomnum',1, ...
        'dl',[9,10]);

    % Integrate
    I2(k)=postint(fem,'nJ_dc', ...
        'solnum',1, ...
        'phase',0*pi/180, ...
        'edim',1, ...
        'intorder',4, ...
        'geomnum',1, ...
        'dl',[15,16]);

    % Integrate
    V1(k)=postint(fem,'V', ...
        'solnum',1, ...
        'phase',0*pi/180, ...
        'edim',1, ...
        'intorder',4, ...
```

```
        'geomnum',1, ...
        'dl',[11,12]);

    % Integrate
    V2(k)=postint(fem,'V', ...
        'solnum',1, ...
        'phase',0*pi/180, ...
        'edim',1, ...
        'intorder',4, ...
        'geomnum',1, ...
        'dl',[13,14]);

   waitbar(k/nSizes, hBar);
end end end

jMean = (I2 - I1)/2; deltaV = V1 - V2; Z = deltaV./ jMean;

close(hBar); toc



% In conducting_layer_1mm_real_solved
function fem = solveFem(thickness,resistivity,size)

% Femlab version
clear vrsn vrsn.name = 'FEMLAB 3.0'; vrsn.ext = 'a'; vrsn.major = 0;
vrsn.build = 228; vrsn.rcs = '$Name:  $'; vrsn.date = '$Date: 2004/04/05
18:04:31 $'; fem.version = vrsn;

% Constants
startPos = sprintf('%d', -size*2);

% Geometry
g1=rect2('200','200','base','corner','pos',{'-100','-200'},'rot','0');
g2=circ2('1','base','center','pos',{startPos,'0'},'rot','0');
garr=geomarrayr(g2,size,0,4,1); [g3,g4,g5,g6]=deal(garr{:}); clear s
s.objs={g1,g2,g4,g5,g6}; s.name={'R1','C1','C2','C3','C4'};
s.tags={'g1','g2','g4','g5','g6'};

fem.draw=struct('s',s);
```

```
% Geometry
g3=geomcomp({g1,g6,g4,g5,g2},'ns',...
{'g1','g6','g4','g5','g2'},'sf','g1-g6-g4-g5-g2','edge','none'); clear s
s.objs={g3}; s.name={'CO1'}; s.tags={'g3'};

fem.draw=struct('s',s); fem.geom=geomcsg(fem);


% Initialize mesh
fem.mesh=meshinit(fem, ...
    'hmax',[], ...
    'hmaxfact',1, ...
    'hgrad',1.3, ...
    'hcurve',0.3, ...
    'hcutoff',0.001, ...
    'hnarrow',1, ...
    'hpnt',10, ...
    'xscale',1.0, ...
    'yscale',1.0, ...
    'report', 'off');

% Constants
rest = resistivity; thick = thickness;

% Application mode 1
clear appl appl.mode.class = 'ConductiveMediaDC'; appl.mode.type =
'cartesian'; appl.dim = {'V'}; appl.sdim = {'x','y','z'}; appl.name =
'dc'; appl.shape = {'shlag(2,''V'')'}; appl.sshape = 2; appl.border =
'off'; appl.assignsuffix = '_dc'; clear prop prop.elemdefault='Lag2';
prop.weakconstr=struct('value',{'off'},'dim',{{'lm1'}}); appl.prop = prop;
clear pnt pnt.Qj0 = 0; pnt.ind = [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1];
appl.pnt = pnt; clear bnd bnd.J0 = {{0;0}}; bnd.sigmabnd = {0,rest,rest};
bnd.d = {1,thick,thick}; bnd.Vref = {0,1,0}; bnd.Jn = 0; bnd.V0 = 0;
bnd.type = {'nJ0','ss','ss'}; bnd.ind = [1,1,1,1,1,1,1,1,2,2,1,1,1,1,3,3];
appl.bnd = bnd; clear equ equ.shape = 1; equ.gporder = 4; equ.cporder = 2;
equ.init = 0; equ.usage = 1; equ.sigma = 1; equ.sigmatensor = 5.99e7;
equ.sigtype = 'iso'; equ.Je = {{0;0}}; equ.Qj = 0; equ.ind = [1]; appl.equ
= equ; fem.appl{1} = appl; fem.sdim = {'x','y'};
```

```
% Simplify expressions
fem.simplify = 'on';

% Global expressions
fem.expr = {};

% Functions
fem.functions = {};

% Multiphysics
fem=multiphysics(fem);

% Extend mesh
fem.xmesh=meshextend(fem,'geoms',[1],'eqvars',...
'on','cplbndeq','on','cplbndsh','off');

% Solve problem
fem.sol=femlin(fem, ...
               'method','eliminate', ...
               'nullfun','flnullorth', ...
               'blocksize',5000, ...
               'complexfun','off', ...
               'conjugate','on', ...
               'symmetric','off', ...
               'solcomp',{'V'}, ...
               'outcomp',{'V'}, ...
               'rowscale','on', ...
               'nonlin','off', ...
               'linsolver','umfpack', ...
               'thresh',0.1, ...
               'umfalloc',0.7, ...
               'uscale','auto', ...
               'report','off');

% Save current fem structure for restart purposes
fem0=fem;
```

G Script for simulating impedance vs length in needle-electrode model

In the calling file *main.m*

```
tic
depth = 30:10:100;

for d = 1:6;
    fem = needle_electrodes(depth(d));

% Integrate
I1(d)=postint(fem,'nJ_dc', ...
           'solnum',1, ...
           'phase',0*pi/180, ...
           'edim',2, ...
           'intorder',4, ...
           'geomnum',1, ...
           'dl',[15,16,25,30]);

% Integrate
I2(d)=postint(fem,'nJ_dc', ...
           'solnum',1, ...
           'phase',0*pi/180, ...
           'edim',2, ...
           'intorder',4, ...
           'geomnum',1, ...
           'dl',[9,10,23,28]);

% Integrate
V3(d)=postint(fem,'V', ...
           'solnum',1, ...
           'phase',0*pi/180, ...
           'edim',2, ...
           'intorder',4, ...
           'geomnum',1, ...
           'dl',[41,42,51,56]);

% Integrate
V4(d)=postint(fem,'V', ...
           'solnum',1, ...
```

```
            'phase',0*pi/180, ...
            'edim',2, ...
            'intorder',4, ...
            'geomnum',1, ...
            'dl',[35,36,49,54]);


       d
end;

deltaV = V4-V3;
I_av = (I2-I1)/2;
Z2 = deltaV./I_av;
toc


function fem = needle_electrodes(x_);


flbinaryfile='needle_electrodes.flm';

% Constants
fem.const={'r','5','s','25','a','s','l','4*a','b','200','d','2*s'};
% Constants
x_loc = sprintf('%d', x_);
x_start = sprintf('%d',-x_/2);

% Geometry
clear draw
g3=flbinary('g3','draw',flbinaryfile);
g4=flbinary('g4','draw',flbinaryfile);
g1=flbinary('g1','draw',flbinaryfile);
g6=flbinary('g6','draw',flbinaryfile);
draw.p.objs = {g3,g4,g1,g6};
draw.p.name = {'PT2','PT3','PT1','PT4'};
draw.p.tags = {'g3','g4','g1','g6'};
g22=flbinary('g22','draw',flbinaryfile);
g10=flbinary('g10','draw',flbinaryfile);
g21=flbinary('g21','draw',flbinaryfile);
g23=flbinary('g23','draw',flbinaryfile);
g25=flbinary('g25','draw',flbinaryfile);
```

```
g12=flbinary('g12','draw',flbinaryfile);
g2=flbinary('g2','draw',flbinaryfile);
g15=flbinary('g15','draw',flbinaryfile);
g11=flbinary('g11','draw',flbinaryfile);
g24=flbinary('g24','draw',flbinaryfile);
g13=flbinary('g13','draw',flbinaryfile);
draw.s.objs = {g22,g10,g21,g23,g25,g12,g2,g15,g11,g24,g13};
draw.s.name = {'CYL7','CYL1','CYL6','CYL8','CYL10',...
'CYL3','BLK1','CYL5','CYL2','CYL9','CYL4'};
draw.s.tags = {'g22','g10','g21','g23','g25','g12',...
'g2','g15','g11','g24','g13'};
fem.draw = draw;
fem.geom = geomcsg(fem);

flbinaryfile='needle_electrodes.flm';

% Geometry
clear p s
p.objs={g3,g4,g1,g6};
p.name={'PT2','PT3','PT1','PT4'};
p.tags={'g3','g4','g1','g6'};

s.objs={g22,g10,g21,g23,g25,g12,g15,g11,g24,g13,g2};
s.name={'CYL7','CYL1','CYL6','CYL8','CYL10','CYL3','CYL5','CYL2',...
'CYL9','CYL4','BLK1'};
s.tags={'g22','g10','g21','g23','g25','g12','g15','g11','g24','g13','g2'};

fem.draw=struct('p',p,'s',s);
fem.geom=geomcsg(fem);

% Geometry
g7=block3('50','100',x_loc,'base','center','pos',{'0','0',x_start},...
'axis',{'0','0','1'},'rot','0');
clear p s
p.objs={g3,g4,g1,g6};
p.name={'PT2','PT3','PT1','PT4'};
p.tags={'g3','g4','g1','g6'};

s.objs={g22,g10,g21,g23,g25,g12,g15,g11,g24,g13,g7};
s.name={'CYL7','CYL1','CYL6','CYL8','CYL10','CYL3','CYL5','CYL2',...
'CYL9','CYL4','BLK1'};
```

```
s.tags={'g22','g10','g21','g23','g25','g12','g15','g11','g24','g13','g7'};

fem.draw=struct('p',p,'s',s);
fem.geom=geomcsg(fem);


% Initialize mesh
fem.mesh=meshinit(fem, ...
                  'hmax',[0], ...
                  'hmaxfact',1, ...
                  'hcutoff',0.01, ...
                  'hgrad',1.4, ...
                  'hcurve',0.4, ...
                  'hnarrow',1, ...
                  'hpnt',20, ...
                  'xscale',1, ...
                  'yscale',1, ...
                  'zscale',1, ...
                  'jiggle','on');

% Application mode 1
clear appl
appl.mode.class = 'ConductiveMediaDC';
appl.mode.type = 'cartesian';
appl.dim = {'V'};
appl.sdim = {'x','y','z'};
appl.name = 'dc';
appl.shape = {'shlag(2,''V'')'};
appl.sshape = 2;
appl.border = 'off';
appl.assignsuffix = '_dc';
clear prop
prop.elemdefault='Lag2';
prop.weakconstr=struct('value',{'off'},'dim',{{'lm1'}});
appl.prop = prop;
clear pnt
pnt.Qj0 = {0,1,-1};
pnt.ind = [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,3,2,1,1,1,1,1,1,...
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1];
appl.pnt = pnt;
clear edg
```

```
edg.Qlj = 0;
edg.ind = [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,...
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,...
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,...
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1];
appl.edg = edg;
clear bnd
bnd.J0 = {{0;0;0}};
bnd.sigmabnd = {0,0,50000000,50000000,0,50000000};
bnd.d = 1;
bnd.Vref = {0,0,-1,1,0,0};
bnd.Jn = {0,0,-1,1,1,0};
bnd.V0 = 0;
bnd.type = {'nJ0','cont','cont','cont','nJ0','cont'};
bnd.ind = [1,1,1,1,1,2,2,2,3,3,2,2,2,2,4,4,2,2,2,2,5,2,3,2,4,2,...
2,3,2,4,2,2,2,2,6,6,2,2,2,2,2,6,6,2,2,2,2,2,1,2,6,2,6,2,2,6,2,6,2,1];
appl.bnd = bnd;
clear equ
equ.shape = 1;
equ.gporder = 4;
equ.cporder = 2;
equ.init = 0;
equ.usage = 1;
equ.sigma = {59900000,1,1e-014};
equ.sigmatensor = 59900000;
equ.sigtype = 'iso';
equ.Je = {{0;0;0}};
equ.Qj = 0;
equ.ind = [2,3,1,3,1,3,3,1,3,1,3];
appl.equ = equ;
fem.appl{1} = appl;
fem.sdim = {'x','y','z'};

% Simplify expressions
fem.simplify = 'on';

% Global expressions
fem.expr = {};

% Functions
fem.functions = {};
```

```
% Multiphysics
fem=multiphysics(fem);

% Extend mesh
fem.xmesh=meshextend(fem,'geoms',[1],'eqvars','on','cplbndeq','on','cplbndsh'


% Solve problem
fem.sol=femlin(fem, ...
               'method','eliminate', ...
               'nullfun','flnullorth', ...
               'blocksize',5000, ...
               'complexfun','off', ...
               'conjugate','on', ...
               'symmetric','on', ...
               'solcomp',{'V'}, ...
               'outcomp',{'V'}, ...
               'rowscale','on', ...
               'nonlin','off', ...
               'linsolver','umfpack', ...
               'thresh',0.1, ...
               'umfalloc',0.7, ...
               'uscale','auto');

% Plot solution
% postplot(fem, ...
%          'slicedata',{'V','cont','internal'}, ...
%          'slicexspacing',[-12.5], ...
%          'sliceyspacing',1, ...
%          'slicezspacing',0, ...
%          'sliceedgestyle','none', ...
%          'slicefacestyle','interp', ...
%          'slicebar','on', ...
%          'slicemap','jet(1024)', ...
%          'solnum',1, ...
%          'phase',0*pi/180, ...
%          'title','Slice: Electric potential', ...
%          'refine',2, ...
%          'geom','on', ...
%          'geomnum',1, ...
```

```
%           'axisvisible','off', ...
%           'axisequal','on', ...
%           'grid','on', ...
%           'camlight','off', ...
%           'scenelight','off', ...
%           'campos',[-603.9867558470161,-787.1308650798779,547.8219618694799], ...
%           'camprojection','orthographic', ...
%           'transparency',1.0);
```

H Complex Materials Script

```
% in main
tic clear omega = power(10,-8:0.5:-3); om = 1:length(omega);

h = waitbar(0,'Computing');

for o = om;
    fem = impedance(omega(o));
    fem_(o) = fem;
% Integrate
V1(o)=postint(fem,'V', ...
            'solnum',1, ...
            'phase',0*pi/180, ...
            'edim',1, ...
            'intorder',4, ...
            'geomnum',1, ...
            'dl',[12]);

% Integrate
V4(o)=postint(fem,'V', ...
            'solnum',1, ...
            'phase',0*pi/180, ...
            'edim',1, ...
            'intorder',4, ...
            'geomnum',1, ...
            'dl',[36]);

% Integrate
V2(o)=postint(fem,'V', ...
            'solnum',1, ...
            'phase',0*pi/180, ...
            'edim',1, ...
            'intorder',4, ...
            'geomnum',1, ...
            'dl',[20]);

% Integrate
V3(o)=postint(fem,'V', ...
            'solnum',1, ...
            'phase',0*pi/180, ...
```

```
            'edim',1, ...
            'intorder',4, ...
            'geomnum',1, ...
            'dl',[28]);

% Integrate
I1(o)=postint(fem,'nJ_dc', ...
            'solnum',1, ...
            'phase',0*pi/180, ...
            'edim',1, ...
            'intorder',4, ...
            'geomnum',1, ...
            'dl',[12]);

% Integrate
I4(o)=postint(fem,'nJ_dc', ...
            'solnum',1, ...
            'phase',0*pi/180, ...
            'edim',1, ...
            'intorder',4, ...
            'geomnum',1, ...
            'dl',[36]);

% Integrate
I2(o)=postint(fem,'nJ_dc', ...
            'solnum',1, ...
            'phase',0*pi/180, ...
            'edim',1, ...
            'intorder',4, ...
            'geomnum',1, ...
            'dl',[20]);

% Integrate
I3(o)=postint(fem,'nJ_dc', ...
            'solnum',1, ...
            'phase',0*pi/180, ...
            'edim',1, ...
            'intorder',4, ...
            'geomnum',1, ...
            'dl',[28]);
```

```
% Integrate
W1(o)=postint(fem,'V2', ...
            'solnum',1, ...
            'phase',0*pi/180, ...
            'edim',1, ...
            'intorder',4, ...
            'geomnum',1, ...
            'dl',[12]);

% Integrate
W4(o)=postint(fem,'V2', ...
             'solnum',1, ...
             'phase',0*pi/180, ...
             'edim',1, ...
             'intorder',4, ...
             'geomnum',1, ...
             'dl',[36]);

% Integrate
W2(o)=postint(fem,'V2', ...
             'solnum',1, ...
             'phase',0*pi/180, ...
             'edim',1, ...
             'intorder',4, ...
             'geomnum',1, ...
             'dl',[20]);

% Integrate
W3(o)=postint(fem,'V2', ...
             'solnum',1, ...
             'phase',0*pi/180, ...
             'edim',1, ...
             'intorder',4, ...
             'geomnum',1, ...
             'dl',[28]);

% Integrate
J1(o)=postint(fem,'nJ_dc2', ...
             'solnum',1, ...
             'phase',0*pi/180, ...
             'edim',1, ...
```

```
            'intorder',4, ...
            'geomnum',1, ...
            'dl',[12]);

% Integrate
J4(o)=postint(fem,'nJ_dc2', ...
            'solnum',1, ...
            'phase',0*pi/180, ...
            'edim',1, ...
            'intorder',4, ...
            'geomnum',1, ...
            'dl',[36]);

% Integrate
J2(o)=postint(fem,'nJ_dc2', ...
            'solnum',1, ...
            'phase',0*pi/180, ...
            'edim',1, ...
            'intorder',4, ...
            'geomnum',1, ...
            'dl',[20]);

% Integrate
J3(o)=postint(fem,'nJ_dc2', ...
            'solnum',1, ...
            'phase',0*pi/180, ...
            'edim',1, ...
            'intorder',4, ...
            'geomnum',1, ...
            'dl',[28]);


    waitbar(log10(o)/length(omega),h);

end;
save results V* W* I* J* fem_;
deltaV = V3-V2; I = I1;
X_real= (real(deltaV./I));
X_imag = (imag(deltaV./I));
close(h);
toc
```

```
% in impedance.m
function fem = impedance(omega)

flbinaryfile='impedance.flm';

% Converting omega to a string
omega = sprintf('%d',omega);


% Constants
fem.const={'w',omega,'eps_skin','1e4','sigma_muscle','.7 +
j*w*eps_muscle','eps_muscle','1e4','sigma_skin','7e-2 +
j*w*eps_skin','eps_fat','30','sigma_fat','1e-2 + j*w*eps_fat'};

% Geometry
clear draw g12=flbinary('g12','draw',flbinaryfile);
g11=flbinary('g11','draw',flbinaryfile); draw.c.objs = {g12,g11};
draw.c.name = {'B2','B1'}; draw.c.tags = {'g12','g11'};
g1=flbinary('g1','draw',flbinaryfile);
g10=flbinary('g10','draw',flbinaryfile);
g9=flbinary('g9','draw',flbinaryfile);
g2=flbinary('g2','draw',flbinaryfile);
g8=flbinary('g8','draw',flbinaryfile); draw.s.objs =
{g1,g10,g9,g2,g8}; draw.s.name = {'SQ1','SQ5','SQ4','SQ2','SQ3'};
draw.s.tags = {'g1','g10','g9','g2','g8'}; fem.draw = draw;
fem.geom = geomcsg(fem);

% Initialize mesh
fem.mesh=meshinit(fem, ...
                  'hmax',[], ...
                  'hmaxfact',1, ...
                  'hgrad',1.3, ...
                  'hcurve',0.3, ...
                  'hcutoff',0.001, ...
                  'hnarrow',1, ...
                  'hpnt',10, ...
                  'xscale',1.0, ...
                  'yscale',1.0,...
                  'report','off');
```

```
% Application mode 1
clear appl appl.mode.class = 'ConductiveMediaDC'; appl.mode.type =
'cartesian'; appl.dim = {'V'}; appl.sdim = {'x','y','z'};
appl.name = 'dc'; appl.shape = {'shlag(2,''V'')'}; appl.sshape =
2; appl.border = 'off'; appl.assignsuffix = '_dc'; clear prop
prop.elemdefault='Lag2';
prop.weakconstr=struct('value',{'off'},'dim',{{'lm1'}}); appl.prop
= prop; clear pnt pnt.Qj0 = 0; pnt.ind =
[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1];
appl.pnt = pnt; clear bnd bnd.J0 = {{0;0}}; bnd.sigmabnd = 0;
bnd.d = 1; bnd.Vref = 0; bnd.Jn =
{0,0,'exp(j*w)','-exp(j*w)','exp(j*w)','-exp(j*w)'}; bnd.V0 = 0;
bnd.type = {'cont','nJ0','nJ0','nJ0','nJ','nJ'}; bnd.ind =
[2,2,2,1,2,1,2,1,1,2,1,5,1,2,2,1,1,2,1,3,1,2,2,1,1,2,1,4,1,2,2,1,1,2,1,6,1,2,2,2,2,2
appl.bnd = bnd; clear equ equ.shape = 1; equ.gporder = 4;
equ.cporder = 2; equ.init = 0; equ.usage = 1; equ.sigma =
{5.99e7,'sigma_skin','sigma_fat','sigma_muscle'}; equ.sigmatensor
= 5.99e7; equ.sigtype = 'iso'; equ.Je = {{0;0}}; equ.Qj = 0;
equ.ind = [4,3,2,1,1,1,1,1,1,1,1]; appl.equ = equ; fem.appl{1} =
appl;

% Application mode 2
clear appl appl.mode.class = 'ConductiveMediaDC'; appl.mode.type =
'cartesian'; appl.dim = {'V2'}; appl.sdim = {'x','y','z'};
appl.name = 'dc2'; appl.shape = {'shlag(2,''V2'')'}; appl.sshape =
2; appl.border = 'off'; appl.assignsuffix = '_dc2'; clear prop
prop.elemdefault='Lag2';
prop.weakconstr=struct('value',{'off'},'dim',{{'lm2'}}); appl.prop
= prop; clear pnt pnt.Qj0 = 0; pnt.ind =
[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1];
appl.pnt = pnt; clear bnd bnd.J0 = {{0;0}}; bnd.sigmabnd = 0;
bnd.d = 1; bnd.Vref = 0; bnd.Jn = {0,0,'exp(j*w)','-exp(j*w)'};
bnd.V0 = 0; bnd.type = {'cont','nJ0','nJ','nJ'}; bnd.ind =
[2,2,2,1,2,1,2,1,1,2,1,2,1,2,2,1,1,2,1,3,1,2,2,1,1,2,1,4,1,2,2,1,1,2,1,2,1,2,2,2,2,2
appl.bnd = bnd; clear equ equ.shape = 1; equ.gporder = 4;
equ.cporder = 2; equ.init = 0; equ.usage = 1; equ.sigma =
{5.99e7,'sigma_muscle'}; equ.sigmatensor = 5.99e7; equ.sigtype =
'iso'; equ.Je = {{0;0}}; equ.Qj = 0; equ.ind =
[2,2,2,1,1,1,1,1,1,1,1]; appl.equ = equ; fem.appl{2} = appl;
fem.sdim = {'x','y'};
```

```
% Simplify expressions
fem.simplify = 'on';

% Global expressions
fem.expr = {'rsens','Real((Jx_dc*Jx_dc2 +
Jy_dc*Jy_dc2)/sigma_dc^2)','isens','Im((Jx_dc*Jx_dc2 +
Jy_dc*Jy_dc2)/sigma_dc^2)'};

% Functions
fem.functions = {};

% Multiphysics
fem=multiphysics(fem);

% Extend mesh
fem.xmesh=meshextend(fem,'geoms',[1],'eqvars','on','cplbndeq','on','cplbndsh'

% Solve problem
fem.sol=femlin(fem, ...
               'method','eliminate', ...
               'nullfun','flnullorth', ...
               'blocksize',5000, ...
               'complexfun','off', ...
               'conjugate','on', ...
               'symmetric','off', ...
               'solcomp',{'V','V2'}, ...
               'outcomp',{'V','V2'}, ...
               'rowscale','on', ...
               'nonlin','off', ...
               'linsolver','umfpack', ...
               'thresh',0.1, ...
               'umfalloc',0.7, ...
               'uscale','auto',...
               'report','off');

% Save current fem structure for restart purposes
fem0=fem;
```